Advanced Cryptography

Serge Vaudenay

EPFL

http://lasec.epfl.ch/

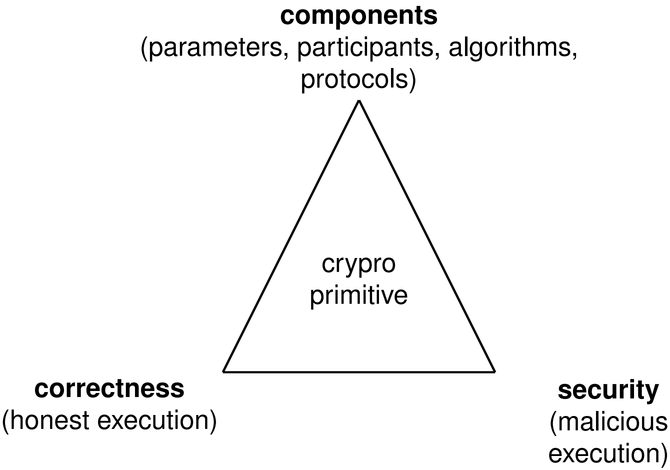
LASEC

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

  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 a 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 $s \to +\infty$, i.e. probabilistic polynomial-time (PPT) algorithm $\to$ 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

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

- example: $f(s) = 2^{-s}$
- avoid writting *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**

Notation: Security Parameter

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

We denote $1^s = \overbrace{11\cdots1}^{s \text{ times}}$, the **unary** representation of s, and consider 1^s as the first input of $\mathcal{A}$:

$$\mathcal{A}(1^s, x; r)$$

Caution: 1^s is <u>often implicit</u> 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. By convention, we prepare “coins”: an infinite sequence of coin flips. $\mathcal{A}$ can read them in sequence whenever needed.

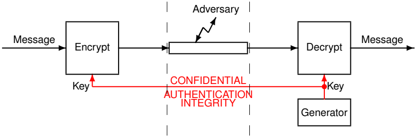
So, $\mathcal{A}$ can also be seen as a **deterministic** function of both input and coins. By convenience, we separate input and coins by a semi-colon: $\mathcal{A}(\text{input}; \text{coins})$

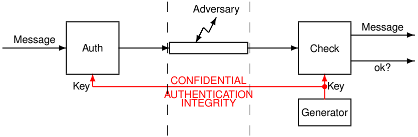
Not all coins are used: there is always a minimal prefix r of coins such that for all sequences s, we have $\mathcal{A}(\text{input}; r\|s) = \mathcal{A}(\text{input}; \text{coins})$. We denote this common value by $\mathcal{A}(\text{input}; r)$.

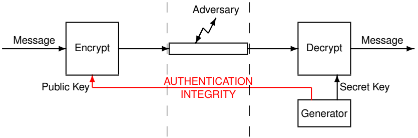
Big Picture

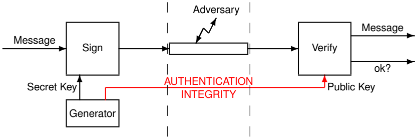
confidential transmission

authenticated transmission









Symmetric Encryption

$$\begin{aligned}\text{encryption} &: \text{key}, \text{nonce}, \text{plaintext} \mapsto \text{ciphertext}\\ \text{decryption} &: \text{key}, \text{nonce}, \text{ciphertext} \mapsto \text{plaintext}\end{aligned}$$

Security goal: protect the confidentiality of plaintexts

- may use a **nonce** (must not be reused in encryption)
- **authenticated encryption**: may use **associated data** as additional input (to authenticate)

$$\begin{aligned}\text{encryption} &: \text{key}, \text{nonce}, \text{ad}, \text{plaintext} \mapsto \text{ciphertext}\\ \text{decryption} &: \text{key}, \text{nonce}, \text{ad}, \text{ciphertext} \mapsto \text{plaintext}\end{aligned}$$

Using Encryption

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

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

- **security**: notion of confidentiality

Using MAC for Authentication

$$\text{MAC} : \text{key}, \text{message} \mapsto \text{tag}$$

- **usage**:

$$\begin{aligned} \text{Auth}_K(X) &= X \| \text{MAC}_K(X) \\ \text{Check}_K(X\|t) &= (X, 1_{t=\text{MAC}_K(X)}) \end{aligned}$$

- **correctness**:

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

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

Public-Key Cryptography

- **public-key cryptosystem**: encryption

$$\begin{aligned}\text{Gen} &: \perp \overset{\$}{\mapsto} \text{pk}, \text{sk} \\ \text{Enc} &: \text{pk}, \text{pt} \overset{\$}{\mapsto} \text{ct} \\ \text{Dec} &: \text{sk}, \text{ct} \mapsto \text{pt}\end{aligned}$$

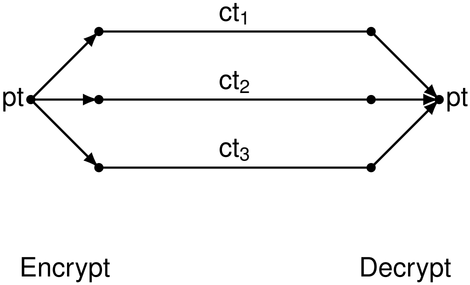
- **digital signature scheme**: authentication + integrity

$$\begin{aligned}\text{Gen} &: \perp \overset{\$}{\mapsto} \text{pk}, \text{sk} \\ \text{Sig} &: \text{sk}, X \overset{\$}{\mapsto} \sigma \\ \text{Ver} &: \text{pk}, X, \sigma \mapsto 0/1\end{aligned}$$

Using Public-Key Encryption

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

Non-Deterministic Encryption



Using Signature

- **usage**:
    - generate a key pair $(\mathsf{pk},\mathsf{sk})$ using Gen
    - the sender keeps sk and publicly reveals pk
    - to sign a message X, compute $\sigma = \mathsf{Sig}(\mathsf{sk},X)$ and send X with σ over the communication channel
    - to validate the message, get (X,σ) and compute $\mathsf{Ver}(\mathsf{pk},X,\sigma)$
- **correctness**: Ver evaluates to 1
- **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**: secrecy of the output K

Commitment Scheme

(most common construction)

Definition

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

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

Using Commitment

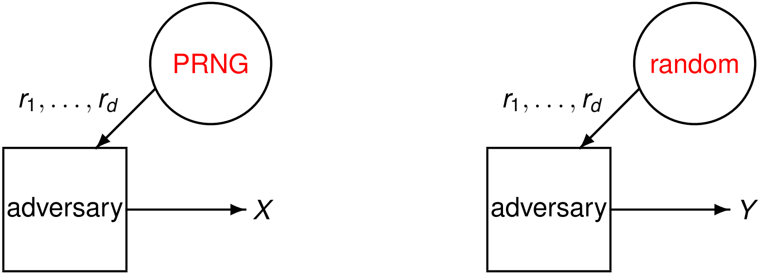
- **usage**:
  - commit to X: pick r and send $c = \text{Commit}(X, r)$
  - open commitment: send X and r; receiver verifies $c = \text{Commit}(X, r)$
- **correctness**: verification succeeds
- **security**: binding and hiding

Other Conventional Primitives

(informal)

- **pseudorandom number generator (PRNG)**
  typically: PRNG(state) $\rightarrow$ (new state, 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

Security of PRNG: Indistinguishability



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

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

Swiss Army Hash Function

- PRNG:

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

- KDF:

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

- 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 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**

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: $\mathbf{Z}_n$, $\mathbf{Z}_p^*$, $\mathrm{GF}(q)^*$, 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 Euclidean division (e.g. $\mathbf{Z}$, $K[x]$)
- **principal ring:** ring in which all ideals have a generator

  $$\text{ring is Euclidean} \Longrightarrow \text{ring is principal}$$
- **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 **Z**:

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

- in $K[x]$:

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

(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 $f : \{1, \ldots, m\} \to \{1, \ldots, n\}$ 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:** $\mathrm{GF}(p^n) = \mathbf{Z}_p[x]/(P(x))$ 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: $\mathbf{Z}_p$, $\mathrm{GF}(2^n)$

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

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

  $$\begin{aligned}\varphi(p_1^{\alpha_1} \cdots p_r^{\alpha_r}) &= (p_1 - 1)p_1^{\alpha_1 - 1} \cdots (p_r - 1)p_r^{\alpha_r - 1} \\ \lambda(p_1^{\alpha_1} \cdots p_r^{\alpha_r}) &= \operatorname{lcm}\left(\lambda(p_1^{\alpha_1}), \ldots, \lambda(p_r^{\alpha_r})\right)\end{aligned}$$

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

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

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 $\mathbf{R}$,

$$V(X) = E\left((X - E(X))^2\right) = E\left(X^2\right) - (E(X))^2$$

- **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 λ, μ:

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

- **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 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**

Arithmetics with Big Numbers

ℓ: 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

ℓ: size of n

- addition ($\mathcal{O}(\ell)$): $x, y, n\ [x, y < n] \mapsto (x + y) \bmod n$
- multiplication ($\mathcal{O}(\ell^2)$): $x, y, n\ [x, y < n] \mapsto (x \times y) \bmod n$
  (Note: asymptotically better algorithm based on FFT, but not better in practice)
- Euclidean division ($\mathcal{O}((\ell + \log x)^2)$): $x, n \mapsto x \bmod n$

- fast exponential ($\mathcal{O}(\ell^2 \log e)$): $x, e, n\ [x < n] \mapsto x^e \bmod n$
  using the **square and multiply algorithm**
- inversion in $\mathbf{Z}_n$ ($\mathcal{O}(\ell^2)$): $x, n\ [x < n] \mapsto y$ s.t. $xy \bmod n = 1$
  (when feasible)
  using the **extended Euclid algorithm**

Other Algorithms

ℓ: size of n

- square root with factorization of $n = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$ ($\mathcal{O}(\ell^3)$):
  $x, p_1^{\alpha_1}, \ldots, p_r^{\alpha_r} \mapsto y$ 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 ($\mathcal{O}(\ell^4)$): $\emptyset \mapsto p$ (ℓ 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	2^n	key length
hash function	preimage attack	2^n	hash length
	collision	$2^{\frac{n}{2}}$	hash length
MAC	key recovery	2^n	key length

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**

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 (polynomialy-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:

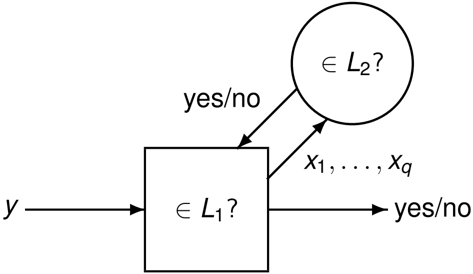
- NFS factoring (factoring n) $e^{\mathcal{O}\left((\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}\right)}$
- ECM factoring (find a small factor p of n) $e^{\mathcal{O}(\sqrt{\ln p \ln\ln p})}$
- GNFS (discrete logarithm in $\mathbf{Z}_p^*$): same as NFS
- Index calculus (discrete logarithm in $\mathbf{Z}_p^*$)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

Turing Reduction

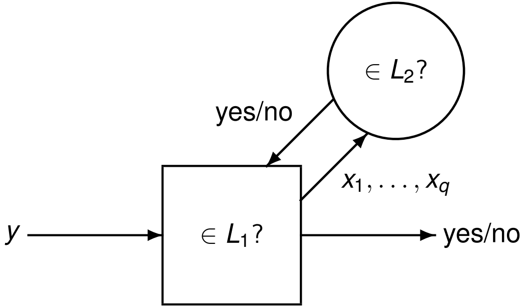
Oracle: Boolean function which says if a word is in a given language. The oracle is connected to a query tape which includes a finite number of non-blank cells.

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 $\Longrightarrow$ L_2 is hard

Conclusion

- **a menagery 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.
- **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 $(\{0,1\}^{k(s)}, \mathcal{D}_s, \mathrm{Enc}, \mathrm{Dec})$ 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

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

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 $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \mathsf{Enc}, \mathsf{Dec})$ with a key domain $\{0,1\}^k$, a plaintext domain $\mathcal{D} = \{X \in \{0,1\}^*; |X| \in \mathcal{L}\}$, a nonce domain $\mathcal{N}$, and two polynomially bounded deterministic algorithms Enc and Dec.
It is such that

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

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

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

```
Game
  $K \xleftarrow{\$} \{0,1\}^k$
  $\mathsf{Used} \leftarrow \emptyset$
  $\mathcal{A}^{\mathsf{OEnc}} \to K'$
  return $1_{K=K'}$
Oracle OEnc($N$, $X$):
  if $N \in \mathsf{Used}$ then return $\perp$    $\triangleright$ nonce-respecting: cannot reuse $N$
  $\mathsf{Used} \leftarrow \mathsf{Used} \cup \{N\}$
  return $\mathsf{Enc}(K, N, X)$
```

Adaptive Security against Key Recovery

Definition

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

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

Game

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

Oracle OEnc(N, X):

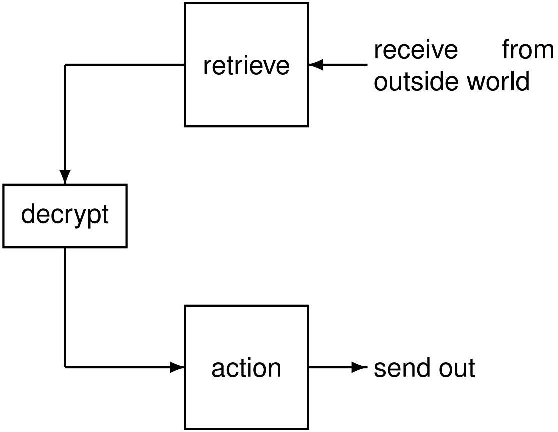
```
5: if $N \in \mathrm{Used}$ then return $\perp$
6: $\mathrm{Used} \leftarrow \mathrm{Used} \cup \{N\}$
7: return $\mathrm{Enc}(K, N, X)$
```

Oracle ODec(N, Y):

```
1: return $\mathrm{Dec}(K, N, Y)$
```

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 CPCA security
- to prove CPA security, consider a CPA adversary $\mathcal{A}$
- we define a CPCA 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 $\mathcal{A}$ is as well □

$$\begin{array}{lcl} \text{CPA-breaking} & \Longrightarrow & \text{CPCA-breaking} \\ \text{CPCA-secure} & \Longrightarrow & \text{CPA-secure} \end{array}$$

Not Good Enough Security

Theorem

We define

- $\mathsf{Enc}(K, N, X) = X$
- $\mathsf{Dec}(K, N, Y) = Y$
- $k = s$

This makes a correct symmetric encryption scheme which is KR-CPCA secure.

Proof. (details on next slide)

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

Not Good Enough Security — Proof

```
$\Gamma_1$:
  $K \xleftarrow{\$} \{0,1\}^k$
  Used $\leftarrow \emptyset$
  $\mathcal{A}^{\text{OEnc},\text{ODec}} \rightarrow K'$
  return $1_{K=K'}$

Oracle OEnc$(N, X)$:
  if $N \in$ Used then return $\perp$
  Used $\leftarrow$ Used $\cup \{N\}$
  return Enc$(K, N, X)$

Oracle ODec$(N, Y)$:
  return Dec$(K, N, Y)$
```

→

```
$\Gamma_2$:
  $K \xleftarrow{\$} \{0,1\}^k$
  Used $\leftarrow \emptyset$
  $\mathcal{A}^{\text{OEnc},\text{ODec}} \rightarrow K'$
  return $1_{K=K'}$

Oracle OEnc$(N, X)$:
  if $N \in$ Used then return $\perp$
  Used $\leftarrow$ Used $\cup \{N\}$
  return $X$

Oracle ODec$(N, Y)$:
  return $Y$
```

↓

```
$\Gamma_3$:
  $K \xleftarrow{\$} \{0,1\}^k$
  $\mathcal{B} \rightarrow K'$
  return $1_{K=K'}$
```

←

- K and K' independent
- K is uniform
- $\Pr[K = K'] = 2^{-k}$

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

Security against Decryption

Definition

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

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

Game

```
$K \xleftarrow{\$} \{0,1\}^k$
$X_0 \xleftarrow{\$} \mathcal{D}$, $N_0 \xleftarrow{\$} \mathcal{N}$
$Y_0 \leftarrow \mathsf{Enc}(K, N_0, X_0)$
$\mathsf{Used} \leftarrow \{N_0\}$
$\mathcal{A}^{\mathsf{OEnc},\mathsf{ODec}}(N_0, Y_0) \rightarrow X$
**return** $1_{X=X_0}$
```

Oracle OEnc(N, X):

```
**if** $N \in \mathsf{Used}$ **then return** $\perp$
$\mathsf{Used} \leftarrow \mathsf{Used} \cup \{N\}$
**return** $\mathsf{Enc}(K, N, X)$
```

Oracle ODec(N, Y):

```
**if** $(N, Y) = (N_0, Y_0)$ **then return** $\perp$
**return** $\mathsf{Dec}(K, N, Y)$
```

Decryption Security is Stronger than Key Recovery Security

- (CPA-security only for simplicity)
- we take a scheme without nonce (for the moment)
- assume we have decryption security and $(\#\mathcal{N})^{-1} = \mathrm{negl}(s)$
- to prove key-recovery security, consider a key recovery adversary $\mathcal{A}$
- we define a decryption adversary $\mathcal{B}$ as follows

```
B(N_0, Y_0) :
 1: run $\mathcal{A} \to \kappa$
    (**if** $\mathcal{A}$ queries $N_0$ **then** abort)   ▷ happens with $\Pr \leq \frac{q}{\#\mathcal{N}}$
 2: compute $X = \mathsf{Dec}(\kappa, N_0, , Y_0)$
 3: **return** $X$
```

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

Detail

- queried nonces by $\mathcal{A}$: $N_1, \ldots, N_q$
- union bound:

$$\begin{aligned}\Pr[\text{abort}] &= \Pr[N_0 = N_1 \vee \cdots N_0 = N_q] \\ &\leq \Pr[N_0 = N_1] + \cdots \Pr[N_0 = N_q]\end{aligned}$$

- $\Pr[N_0 = N_i] = \frac{1}{\#\mathcal{N}}$
- $\Pr[\mathcal{B}\text{ wins}] + \Pr[\text{abort}] \geq \Pr[\mathcal{A}\text{ wins}]$

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

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

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

The Ideal Cipher

- the ideal block cipher: $(\mathsf{Perm}(\{0,1\}^n), \{0,1\}^n, \mathsf{Enc}, \mathsf{Dec})$

$$\mathsf{Enc}(\Pi, X) = \Pi(X) \qquad \mathsf{Dec}(\Pi, Y) = \Pi^{-1}(Y)$$

- the "ideal cipher": taking K random is equivalent to picking a random length-preserving permutation Π_N for every N

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

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

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

Game Γ_b

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

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

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

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

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

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

```
$\mathcal{B}$ :
  $X_0 \xleftarrow{\$} \mathcal{D}$, $N_0 \xleftarrow{\$} \mathcal{N}$
  $Y_0 \leftarrow \mathsf{OEnc}(N_0, X_0)$
  run $\mathcal{A}^{\mathsf{OEnc},\mathsf{ODec}'}(N_0, Y_0) \rightarrow X$
  return $1_{X_0 = X}$
```

```
$\mathsf{ODec}'(N, Y)$ :
  if $(N, Y) = (N_0, Y_0)$ then return $\perp$
  return $\mathsf{ODec}(N, Y)$
```

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

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

Security Notions

	key recovery	**decryption**	**distinguisher**
CPA	weakest security		
CPCA			strongest security

- if we can recover the key, we can decrypt
- if we can decrypt, we can tell ciphers aparts
- if we can break without chosen ciphertext, we can also break with

Note: Another Distinguisher Style

Game Γ':

```
1: pick $b \in \{0,1\}$
2: run $\Gamma_b \to z$
3: **return** $1_{b=z}$
```

$$\mathsf{Adv}^{\Gamma'} = \Pr[\Gamma' \to 1] - \tfrac{1}{2}$$

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

MAC

(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 <u>deterministic</u> algorithm MAC implementing a function

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

Using MAC for Authentication

- **usage**:

$$\begin{aligned}\mathsf{Auth}_K(X) &= X\|\mathsf{MAC}_K(X)\\ \mathsf{Check}_K(X\|t) &= (X, 1_{t=\mathsf{MAC}_K(X)})\end{aligned}$$

- **correctness**:

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

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

Security against Key Recovery

Definition

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

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

```
Game
  1: $K \xleftarrow{\$} \{0,1\}^k$
  2: $\mathcal{A}^{\mathsf{OMac}} \to K'$
  3: return $1_{K=K'}$

Oracle OMac($X$):
  4: return $\mathsf{MAC}(K, X)$
```

Security against Forgery

Definition

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

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

```
Game
1: $K \xleftarrow{\$} \{0,1\}^k$
2: $\mathsf{Queried} \leftarrow \emptyset$
3: $\mathcal{A}^{\mathsf{OMac}} \to (X, t)$
4: if $X \in \mathsf{Queried}$ then
   return 0
5: return $1_{\mathsf{MAC}(K,X)=t}$
```

```
Oracle OMac(X):
6: $\mathsf{Queried} \leftarrow \mathsf{Queried} \cup \{X\}$
7: return $\mathsf{MAC}(K, X)$
```

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: $\mathcal{A}^{\mathrm{OMac}}(X_0) \rightarrow t$
  4: return $1_{\mathrm{MAC}(K,X_0)=t}$
  ```

  Oracle OMac(X):

  ```
  1: if $X = X_0$ then return $\perp$
  2: return $\mathrm{MAC}(K,X)$
  ```

- **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 Adv is negligible.

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

```
Game $\Gamma_b$
  $K \xleftarrow{\$} \{0,1\}^k$
  pick $F^* : \mathcal{D} \to \{0,1\}^\tau$
  $\mathcal{A}^{\mathsf{O}} \to z$
  return $z$
```

```
Oracle $\mathsf{O}(X)$:
  if $b = 0$ then return $F^*(X)$
  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 $\Longrightarrow$ UF-secure $\Longrightarrow$ key recovery-secure
- $(\text{PRF} \wedge 2^{-\tau} = \text{negl}) \Longrightarrow$ 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**: $\mathsf{Adv} = \Pr[\text{Game returns } 1]$

  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**:

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

  Game Γ_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 $z$
  ```

PKC

Definition

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

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

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

Threat Models

Adversary capability:

- **CPA**: encrypt chosen plaintexts (always: enc key is public)
- **CCA**: access to a decryption oracle

Adversary goal:

- **Key recovery (KR)**: recover the secret key
- **Decryption (OW)**: decrypt a *random* ciphertext
- **Information recovery**: infer a given bit of information on the plaintext given a random ciphertext
- **Distinguishability (IND)**: 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 $(\mathsf{Gen}, \mathcal{M}, \mathsf{Enc}, \mathsf{Dec})$ if for any PPT adversary $\mathcal{A}$, the advantage Adv is negligible.

$$\mathsf{Adv} = 2\Pr[\text{game returns } 1] - 1$$

Game

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

Security against Distinguisher (Left or Right)

Definition

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

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

Game Γ_b

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

Remark: Two Styles of Interactive Adversaries

Two algorithms with state variable

```
Game $\Gamma_b$
  $\mathsf{Gen} \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
  $\mathcal{A}_1(\mathsf{pk}) \xrightarrow{\$} (\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$
  if $|\mathsf{pt}_0| \neq |\mathsf{pt}_1|$ then return $0$
  $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_b)$
  $\mathcal{A}_2(\mathsf{st}, \mathsf{ct}) \xrightarrow{\$} z$
  return $z$
```

Deterministic algorithm with variable number of inputs

```
Game $\Gamma_b$
  pick $\rho$ at random
  $\mathsf{Gen} \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
  $\mathcal{A}(\mathsf{pk}; \rho) \to (\mathsf{pt}_0, \mathsf{pt}_1)$
  if $|\mathsf{pt}_0| \neq |\mathsf{pt}_1|$ then return $0$
  $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_b)$
  $\mathcal{A}(\mathsf{pk}, \mathsf{ct}; \rho) \to z$
  return $z$
```

Problem with Deterministic Cryptosystems

- IND-CPA is a modern notion of security
- problem: if 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 $(\mathsf{Gen}, \mathcal{M}, \mathsf{Enc}, \mathsf{Dec})$ is **secure under chosen ciphertext attacks** (IND-CCA-secure) if for any interactive PPT process $(\mathcal{A}_1, \mathcal{A}_2)$, the advantage Adv is negligible.

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

```
Game $\Gamma_b$
  $\mathsf{Gen} \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
  $\mathcal{A}_1^{\mathsf{ODec}_1}(\mathsf{pk}) \xrightarrow{\$} (\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$
  if $|\mathsf{pt}_0| \neq |\mathsf{pt}_1|$ then return 0
  $\mathsf{ct}^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_b)$
  $\mathcal{A}_2^{\mathsf{ODec}_2}(\mathsf{st}, \mathsf{ct}^*) \xrightarrow{\$} z$
  return $z$
```

```
Oracle $\mathsf{ODec}_1(\mathsf{ct})$:
  return $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
```

```
Oracle $\mathsf{ODec}_2(\mathsf{ct})$:
  if $\mathsf{ct} = \mathsf{ct}^*$ then return $\perp$
  return $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
```

Non-Malleability

Definition (Malleability (intuitively))

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

	key recovery	decryption	distinguisher
CPA	weakest security		
CCA			strongest security

- if we can recover the key, we can decrypt
- if we can decrypt, we can tell ciphers apart
- if we can break without chosen ciphertext, we can also break with

Signature Scheme

Definition

A **digital signature scheme** is a tuple $(\mathsf{Gen}, \mathcal{D}, \mathsf{Sig}, \mathsf{Ver})$ 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

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

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

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

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

```
Game
  $\mathsf{Gen} \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
  $\mathsf{Queries} \leftarrow \emptyset$
  $\mathcal{A}^{\mathsf{OSig}}(\mathsf{pk}) \rightarrow (X, \sigma)$
  if $X \in \mathsf{Queries}$ then
    return 0
  return $1_{\mathsf{Ver}(\mathsf{pk}, X, \sigma)}$

Oracle OSig($X$):
  $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, X)$
  $\mathsf{Queries} \leftarrow \mathsf{Queries} \cup \{X\}$
  return $\sigma$
```

(Strong) EF-CMA Security

Definition

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

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

```
Game
  Gen $\xrightarrow{\$}$ (pk, sk)
  Queries $\leftarrow \emptyset$
  $\mathcal{A}^{\mathsf{OSig}}(\mathsf{pk}) \rightarrow (X, \sigma)$
  if $(X, \sigma) \in$ Queries then
      return 0
  return $1_{\mathsf{Ver}(\mathsf{pk}, X, \sigma)}$
```

```
Oracle OSig($X$):
  $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, X)$
  Queries $\leftarrow$
      Queries $\cup \{(X, \sigma)\}$
  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			(default)
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

$$\begin{tikzcd}
\text{game} & & \text{adversary} \\
\boxed{\Gamma_1} \arrow[d] & & \mathcal{A} \arrow[ll] \arrow[d] \\
\boxed{\Gamma_2} & & C_2(\mathcal{A}) \arrow[ll] \\
\vdots & & \vdots \\
\boxed{\Gamma_{n-1}} \arrow[d] & & C_{n-1}(\mathcal{A}) \arrow[ll] \arrow[d] \\
\boxed{\Gamma_n} & & C_n(\mathcal{A}) \arrow[ll]
\end{tikzcd}$$

$$\Pr[\Gamma_1(\mathcal{A}) = 1] - \Pr[\Gamma_2(C_2(\mathcal{A})) = 1] \leq \varepsilon_2$$

$$\vdots$$

$$\Pr[\Gamma_{n-1}(C_{n-1}(\mathcal{A})) = 1] - \Pr[\Gamma_n(C_n(\mathcal{A})) = 1] \leq \varepsilon_n$$

$$\varepsilon_2 + \cdots + \varepsilon_n = \text{negl and } \Pr[\Gamma_n(C_n(\mathcal{A})) = 1] = \text{negl}$$

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 Γ
- F: a ("failure") event in Γ
- (Γ becomes "simpler" when $\neg F$ holds)
- define Γ' like Γ with $\neg F$ as an extra winning condition

```
$\Gamma'(\mathcal{A})$:
  $\Gamma(\mathcal{A}) \rightarrow z$ (check if $F$ holds)
  **return** $1_{z=1\wedge\neg F}$
```

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

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

Γ' is a kind of rewriting of Γ

Transition Tool 3: Bridging Step Examples

- case 0: permute two operations which are independent
- case 1: move operations between Γ 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 $T$
if $T(x)$ is defined then
    $y \leftarrow T(x)$
else
    pick $y$ at random
    $T(x) \leftarrow y$
end if
return $y$
```

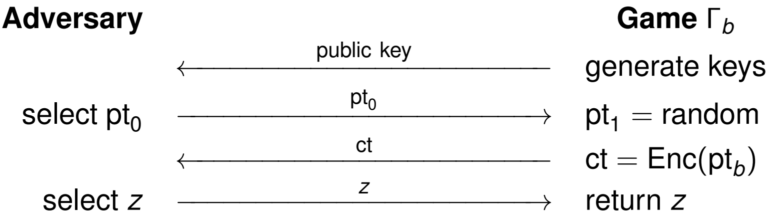
Double Bridge

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

$$\begin{array}{l}\Gamma_0(\mathcal{A}):\\ \ 1:\ \dots\\ \ 2:\ \textbf{return}\ b\end{array} \overset{\text{bridge}}{\frown} \begin{array}{l}\Gamma'_0(C(\mathcal{A})):\\ \ 1:\ \gamma_0 \to z\\ \ 2:\ C(\mathcal{A})(z) \to b'\\ \ 3:\ \textbf{return}\ b'\end{array} \overset{\text{tool1}}{\approx} \begin{array}{l}\Gamma'_1(C(\mathcal{A})):\\ \ 1:\ \gamma_1 \to z\\ \ 2:\ C(\mathcal{A})(z) \to b'\\ \ 3:\ \textbf{return}\ b'\end{array} \overset{\text{bridge}}{\frown} \begin{array}{l}\Gamma_1(\mathcal{A}):\\ \ 1:\ \dots\\ \ 2:\ \textbf{return}\ b\end{array}$$

$$\begin{tikzcd} \Gamma_0 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma'_0 \arrow[d, leftrightarrow, "\text{distinguisher between } \gamma_0 \text{ and } \gamma_1 \text{ (tool 1)}"] \\ \Gamma_1 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma'_1 \end{tikzcd}$$

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



Adversary

Game Γ_b

public key

generate keys

select pt_0

pt_0

$\mathsf{pt}_1 = \text{random}$

ct

$\mathsf{ct} = \mathsf{Enc}(\mathsf{pt}_b)$

select z

z

return z

Game Γ_b

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

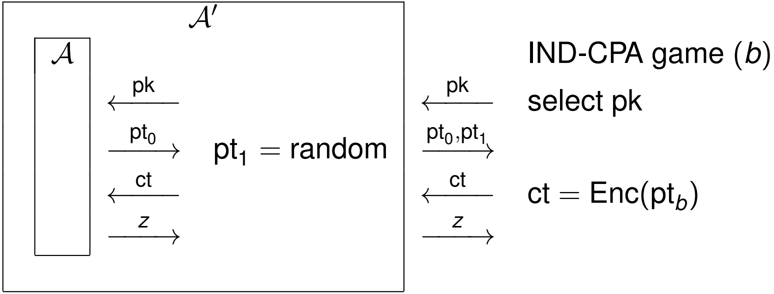
$\mathcal{A}$ works in two steps: $\mathcal{A}_1$, $\mathcal{A}_2$
a state variable st is kept

$$\mathsf{Adv} = \Pr[\Gamma_1 \to 1] - \Pr[\Gamma_0 \to 1]$$

Equivalence with IND-CPA Game — i

IND-CPA secure $\Longrightarrow$ IND\$-CPA secure:

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


$\mathcal{A}'$
$\mathcal{A}$
pk
pt_0
ct
z
$\text{pt}_1 = \text{random}$
pk
pt_0, pt_1
ct
z
IND-CPA game (b)
select pk
$\text{ct} = \text{Enc}(\text{pt}_b)$

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

Equivalence with IND-CPA Game — i (bis)

- Γ_b: $\mathcal{A}$ playing IND\$-CPA
- Γ'_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

$$\begin{tikzcd}
\Gamma_0 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma'_0 \arrow[d, leftrightarrow, "\text{indistinguishable due to IND-CPA security}"] \\
\Gamma_1 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma'_1
\end{tikzcd}$$

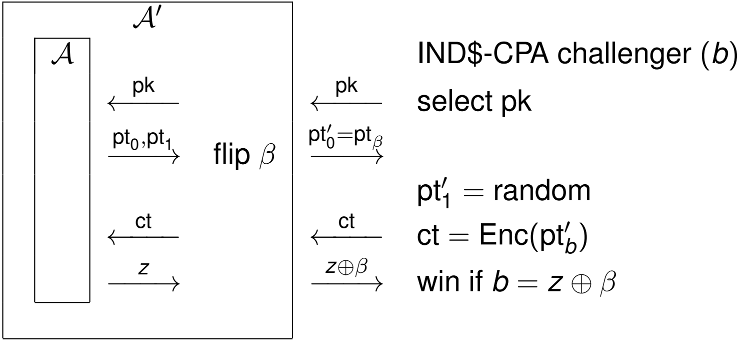
Equivalence with IND-CPA Game — i (ter)

$$
\begin{array}{ccc}
\text{IND\$-CPA}_b(\mathcal{A}) & & \text{IND-CPA}_b(\mathcal{A}') \\[1em]
\Pr\left[\left[\begin{array}{l}
\mathsf{Gen} \to (\mathsf{pk}, \mathsf{sk}) \\
\mathcal{A}_1(\mathsf{pk}) \to (\mathsf{pt}_0, \mathsf{st}) \\
\text{pick } \mathsf{pt}_1 \\
\mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_0) \to \mathsf{ct} \\
\mathcal{A}_2(\mathsf{st}, \mathsf{ct}) \to z \\
\text{return } z
\end{array}\right] \to 1\right]
& \overset{\text{bridge}}{=} &
\Pr\left[\left[\begin{array}{l}
\mathsf{Gen} \to (\mathsf{pk}, \mathsf{sk}) \\
\mathcal{A}_1(\mathsf{pk}) \to (\mathsf{pt}_0, \mathsf{st}) \\
\text{pick } \mathsf{pt}_1 \\
\mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_0) \to \mathsf{ct} \\
\mathcal{A}_2(\mathsf{st}, \mathsf{ct}) \to z \\
\text{return } z
\end{array}\right] \to 1\right] \\
 & & \wr\ (\text{IND-CPA}) \\
\Pr\left[\left[\begin{array}{l}
\mathsf{Gen} \to (\mathsf{pk}, \mathsf{sk}) \\
\mathcal{A}_1(\mathsf{pk}) \to (\mathsf{pt}_0, \mathsf{st}) \\
\text{pick } \mathsf{pt}_1 \\
\mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_1) \to \mathsf{ct} \\
\mathcal{A}_2(\mathsf{st}, \mathsf{ct}) \to z \\
\text{return } z
\end{array}\right] \to 1\right]
& \overset{\text{bridge}}{=} &
\Pr\left[\left[\begin{array}{l}
\mathsf{Gen} \to (\mathsf{pk}, \mathsf{sk}) \\
\mathcal{A}_1(\mathsf{pk}) \to (\mathsf{pt}_0, \mathsf{st}) \\
\text{pick } \mathsf{pt}_1 \\
\mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_1) \to \mathsf{ct} \\
\mathcal{A}_2(\mathsf{st}, \mathsf{ct}) \to z \\
\text{return } z
\end{array}\right] \to 1\right]
\end{array}
$$

Equivalence with IND-CPA Game — ii

IND\$-CPA secure $\Longrightarrow$ IND-CPA secure:

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



$\mathcal{A}'$

$\mathcal{A}$

IND\$-CPA challenger (b)

$\xleftarrow{\text{pk}}$ $\xleftarrow{\text{pk}}$ select pk

$\xrightarrow{\text{pt}_0,\text{pt}_1}$ flip β $\xrightarrow{\text{pt}'_0=\text{pt}_\beta}$

$\text{pt}'_1 = \text{random}$

$\xleftarrow{\text{ct}}$ $\xleftarrow{\text{ct}}$ $\text{ct} = \text{Enc}(\text{pt}'_b)$

$\xrightarrow{z}$ $\xrightarrow{z\oplus\beta}$ win if $b = z \oplus \beta$

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

Equivalence with IND-CPA Game — ii (bis)

$\text{IND\$-CPA}_{b=1}(\mathcal{A}')$

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

Equivalence with IND-CPA Game — ii (ter)

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

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

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

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

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

1. The Cryptographic Zoo

2. **Cryptographic Security Models**
   - Security Definitions
   - The Game Proof Methodology
   - RSA Security
   - Rabin Cryptosystem
   - Diffie-Hellman Security
   - ElGamal Security

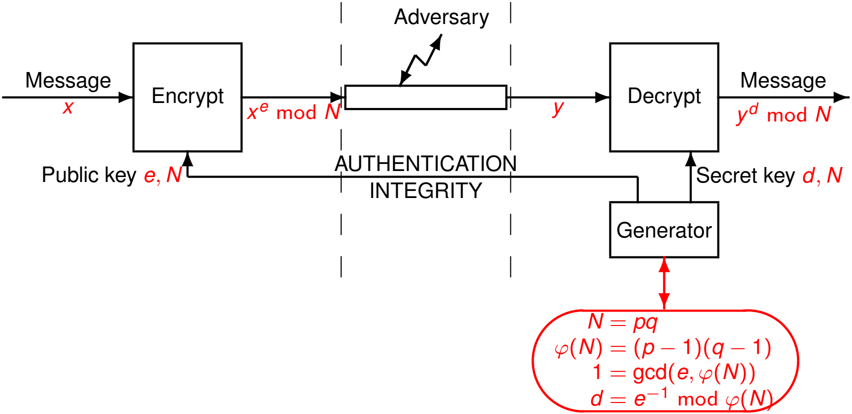
3. Cryptanalysis (Public-Key)

4. The Power of Interaction

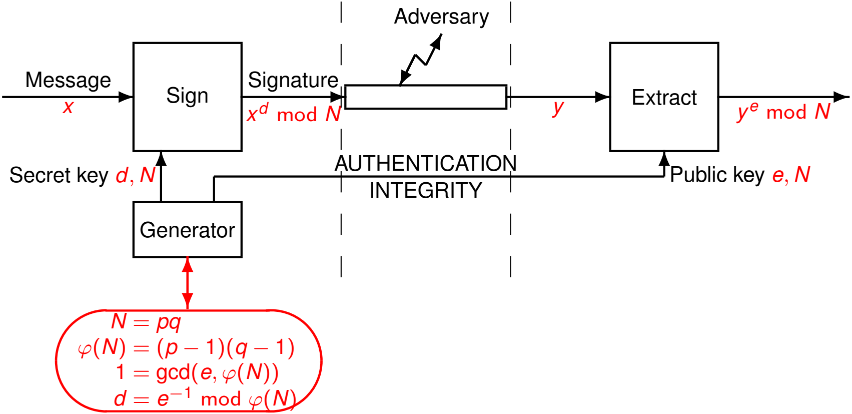
5. Cryptanalysis (Conventional)

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

RSA Factoring

Factor N generated by Gen:

Game

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

RSA assumption: the RSA problem is hard

Bit Security of Plain RSA

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

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

Theorem

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

(even approximations...)

Reducing Decryption to lsbdec

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

- 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 β s.t. k replaced by $2k + \beta$
- end up with $a \leq x < b$ with $b - a \leq 1$

```
$a \leftarrow 0$, $b \leftarrow N$
for $i = 0$ to $\lfloor \log_2 N \rfloor$ do
    if lsbdec($2^{(i+1)e} y \bmod N$) $= 1$ then
        $a \leftarrow (a+b)/2$
    else
        $b \leftarrow (a+b)/2$
    end if
end for
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)

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

which is easy to compute even without the factorization of N

Reminder: $x \mapsto (x/N)$ is an easy-to-compute homomorphism from $\mathbf{Z}_N^*$ to $\{-1,+1\}$.

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 OW-CCA security: $\mathsf{Dec}(y \cdot u^e \bmod N)/u = \mathsf{Dec}(y)$

Strong RSA Problem

(implicit: Gen is the RSA key generation algorithm)

Strong RSA Problem

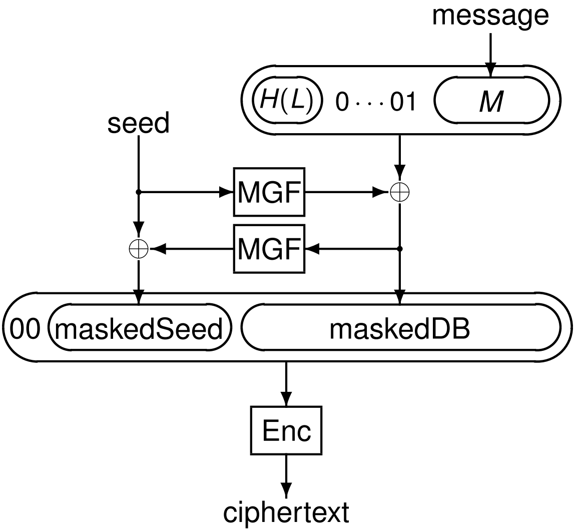
Given N generated by Gen and a random residue y, compute x and $e > 1$ such that $y = x^e \bmod N$:

Game

```
1: $\mathrm{Gen}(1^s) \xrightarrow{\$} N$
2: pick $y \in \mathbf{Z}_N$
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


message
$H(L)$
$0\cdots01$
M
seed
MGF
$\oplus$
MGF
$\oplus$
00
maskedSeed
maskedDB
Enc
ciphertext

Security of RSA-OAEP

- Security results are far beyond this lecture
- They exist for the IND-CCA notion (in the random oracle model, under the RSA assumption)
- Its significance was controversial
  (The original proof was wrong.)

1. The Cryptographic Zoo

2. **Cryptographic Security Models**
   - Security Definitions
   - The Game Proof Methodology
   - RSA Security
   - Rabin Cryptosystem
   - Diffie-Hellman Security
   - ElGamal Security

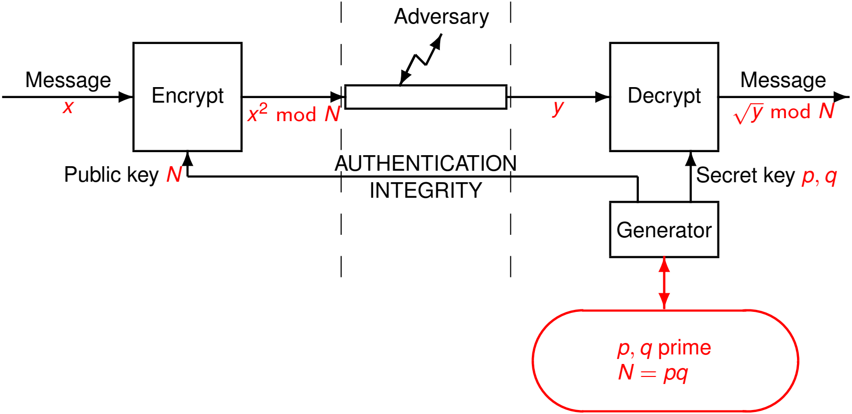
3. Cryptanalysis (Public-Key)

4. The Power of Interaction

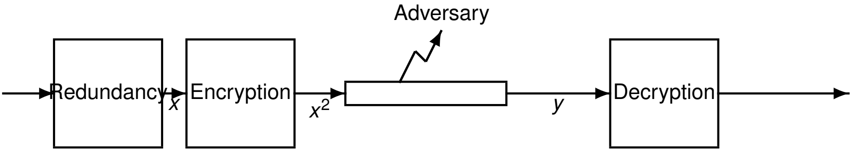
5. Cryptanalysis (Conventional)

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

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

OW-CPA (Rabin Decryption)

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

Game

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

KR-CPA (Rabin Key Recovery)

Factor N generated by Gen:

Game

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

decryption problem	$\Longleftrightarrow$	factorization problem
key recovery problem	$\Longleftrightarrow$	factorization problem

Factoring Hard $\Longrightarrow$ OW-CPA Security

```
Game OW-CPA
Gen($1^s$) $\xrightarrow{\$}$ $N$
pick $x \in \mathbf{Z}_N$
$y \leftarrow x^2 \bmod N$
$\mathcal{A}(N, y) \xrightarrow{\$} z$
return $1_{z=x}$
```

→

```
Game Γ
Gen($1^s$) $\xrightarrow{\$}$ $N$
pick $x \in \mathbf{Z}_N^*$
$y \leftarrow x^2 \bmod N$
$\mathcal{A}(N, y) \xrightarrow{\$} z$
return $1_{z=x}$
```

↓

```
Game Fact
Gen($1^s$) $\xrightarrow{\$}$ $N$
$\mathcal{B}(N) \xrightarrow{\$} (p, q)$
return $1_{1<p<N, N=pq}$

$\mathcal{B}(N)$:
pick $x \in \mathbf{Z}_N^*$
$y \leftarrow x^2 \bmod N$
$\mathcal{A}(N, y) \xrightarrow{\$} z$
$p \leftarrow \gcd(z - x, N)$
if $p \in \{1, N\}$ then abort
$q \leftarrow N/p$
return $(p, q)$
```

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

$|\Pr[\text{OW-CPA} \rightarrow 1] - \Pr[\Gamma \rightarrow 1]| \leq \varepsilon$ (difference lemma)

when $z^2 \equiv y$, x is indep. uniform in SQRT(y) (4 elements)

$\Pr[\Gamma \rightarrow 1 | z^2 \equiv y] = \frac{1}{4}$

$\Pr[\text{Fact} \rightarrow 1 | z^2 \equiv y] = \frac{1}{2}$

$$\begin{aligned} &\Pr[\Gamma \rightarrow 1] \\ &= \Pr[\Gamma \rightarrow 1, z^2 = y] \\ &= \tfrac{1}{2} \Pr[\text{Fact} \rightarrow 1, z^2 = y] \\ &\leq \tfrac{1}{2} \Pr[\text{Fact} \rightarrow 1] = \text{negl} \end{aligned}$$

A KR-CCA Attack against Rabin

```
Game KR-CCA
    $\mathsf{Gen}(1^s) \xrightarrow{\$} \mathsf{sk}, N$
    $\mathcal{B}^{\mathsf{ODec}}(N) \xrightarrow{\$} (p, q)$
    return $1_{(p,q)=\mathsf{sk}}$

$\mathcal{B}(N)$:
    pick $x \in \mathbf{Z}_N^*$
    $y \leftarrow x^2 \bmod N$
    $\mathsf{ODec}(y) \rightarrow z$
    $p \leftarrow \gcd(z - x, N)$
    if $p \in \{1, N\}$ then abort
    $q \leftarrow N/p$
    return $(p, q)$
```

```
ODec($y$):
    compute a square root of $y$ in $\mathbf{Z}_N$ using sk
    return result
```

Paradoxical Security Result

- Rabin cryptosystem is PROVABLY as secure (OW-CPA) as the factorization is hard
- the security proof yields a CHOSEN CIPHERTEXT ATTACK (KR-CCA)

- 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 ($\mathbf{Z}_p^*$, an elliptic curve, ...) generated by some g

$$\begin{array}{rcl}
\textbf{Alice} & & \textbf{Bob} \\
\text{pick } x \text{ at random, } X \leftarrow g^x & \xrightarrow{\quad X \quad} & \\
& \xleftarrow{\quad Y \quad} & \text{pick } y \text{ at random, } Y \leftarrow g^y \\
K \leftarrow Y^x & & K \leftarrow X^y \\
& (K = g^{xy}) &
\end{array}$$

resists passive adversaries

CDH vs DL Problems

implicit: Setup → (group, q, g)

- group: parameters to perform group operations
- q: integer (order of the group)
- g: group element (generator)

CDH (Computational Diffie-Hellman)

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

Game

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ q, g$
pick $x, y \in \mathbf{Z}_q$
$X \leftarrow g^x$, $Y \leftarrow g^y$
$\mathcal{A}(\mathsf{pp}, X, Y) \xrightarrow{\$} K$
**return** $1_{K = g^{xy}}$
```

DL (Discrete Logarithm)

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

Game

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ q, g$
pick $x \in \mathbf{Z}_q$
$X \leftarrow g^x$
$\mathcal{A}(\mathsf{pp}, X) \xrightarrow{\$} z$
**return** $1_{X = g^z}$
```

CDH is the Key Recovery Problem in DH

```
Game CDH
1: Setup($1^s$) $\xrightarrow{\$}$ pp        $\triangleright$ $q, g$
2: pick $x, y \in \mathbf{Z}_q$
3: $X \leftarrow g^x$, $Y \leftarrow g^y$
4: $\mathcal{A}(\mathrm{pp}, X, Y) \xrightarrow{\$} K$
5: return $1_{K=g^{xy}}$
```

```
Game KR with DH
1: Setup($1^s$) $\xrightarrow{\$}$ pp        $\triangleright$ $q, g$
2: execute $A$(pp) $\leftrightarrow$ $B$(pp)
3: get transcript and output $K$
4: run $\mathcal{A}(\mathrm{pp}, \mathrm{transcript}) \xrightarrow{\$} K'$
5: return $1_{K=K'}$
```

CDH Hard $\Longrightarrow$ DL Hard

Pr
```
Game DL
  1: $\text{Setup}(1^s) \xrightarrow{\$} \text{pp}$        ▷ $q, g$
  2: pick $x \in \mathbf{Z}_q$
  3: $X \leftarrow g^x$
  4: $\mathcal{A}(\text{pp}, X) \xrightarrow{\$} z$
  5: return $1_{X=g^z}$
```

$\leq$

Pr
```
  1: $\text{Setup}(1^s) \xrightarrow{\$} \text{pp}$        ▷ $q, g$
  2: pick $x, y \in \mathbf{Z}_q$
  3: $X \leftarrow g^x$
  4: $\mathcal{A}(\text{pp}, X) \xrightarrow{\$} z$
  5: return $1_{X^y=g^{yz}}$
```

$\shortparallel$

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

$\mathcal{B}(\text{pp}, X, Y)$:
  6: $\mathcal{A}(\text{pp}, X) \xrightarrow{\$} z$
  7: $K \leftarrow Y^z$
  8: return $K$
```

$=$

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

Decisional DH Problem

DDH (Decisional Diffie-Hellman)

$$\mathsf{Adv} = \Pr[\Gamma_1 \to 1] - \Pr[\Gamma_0 \to 1]$$

Game Γ_b

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ q, g$
pick $x, y \in \mathbf{Z}_q$
if $b = 0$ then
    pick $z \in \mathbf{Z}_q$
else
    $z \leftarrow xy \bmod q$
end if
$X \leftarrow g^x$, $Y \leftarrow g^y$, $Z \leftarrow g^z$
$\mathcal{A}(\mathsf{pp}, X, Y, Z) \xrightarrow{\$} c$
return $c$
```

DDH is the key distinguiher problem with DH

DDH is the Key Distinguisher Problem in DH

```
Game DDH$_b$
    Setup$(1^s) \xrightarrow{\$}$ pp        $\triangleright\ q, g$
    pick $x, y \in \mathbf{Z}_q$
    $z \leftarrow b = 0$ ? random : $xy$
    $X \leftarrow g^x$, $Y \leftarrow g^y$, $Z \leftarrow g^z$
    $\mathcal{A}(\mathrm{pp}, X, Y, Z) \xrightarrow{\$} c$
    **return** $c$
```

```
Game KD with DH
    Setup$(1^s) \xrightarrow{\$}$ pp        $\triangleright\ q, g$
    execute $A(\mathrm{pp}) \leftrightarrow B(\mathrm{pp})$
    get transcript and $K_1$
    set $K_0$ at random
    $\mathcal{A}(\mathrm{pp}, \mathrm{transcript}, K_b) \xrightarrow{\$} z$
    **return** $z$
```

DDH Hard $\Longrightarrow$ CDH Hard

Game CDH

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ q, g$
pick $x, y \in \mathbf{Z}_q$
$X \leftarrow g^x,\ Y \leftarrow g^y$
$\mathcal{A}(\mathsf{pp}, X, Y) \xrightarrow{\$} K$
**return** $1_{K=g^{xy}}$
```

Game DDH

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ q, g$
pick $x, y \in \mathbf{Z}_q$
$z \leftarrow b = 0\ ?\ \mathsf{random} : xy$
$X \leftarrow g^x,\ Y \leftarrow g^y,\ Z \leftarrow g^z$
$\mathcal{B}(\mathsf{pp}, X, Y, Z) \xrightarrow{\$} c$
**return** $c$
```

$\mathcal{B}(\mathsf{pp}, X, Y, Z)$:

```
$\mathcal{A}(\mathsf{pp}, X, Y) \xrightarrow{\$} K$
**return** $1_{K=Z}$
```

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

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

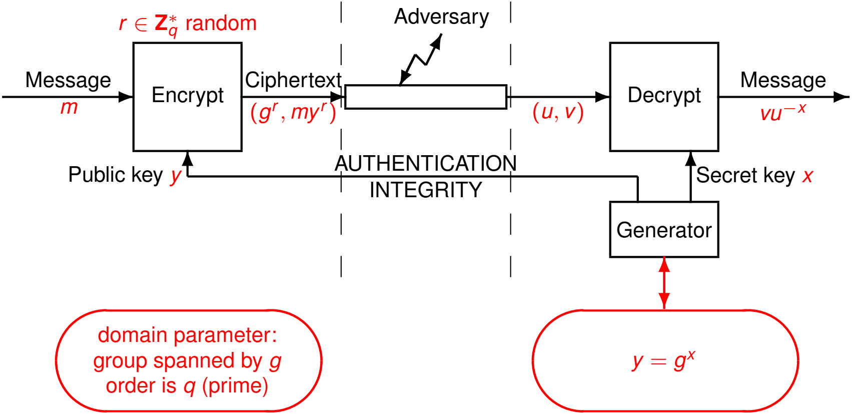
DDH Hard $\Longrightarrow$ Large Group

```
$\mathcal{B}'(\mathrm{pp}, X, Y, Z)$:
  pick $x$ at random
  **if** $X = g^x$ **then**
      **if** $Z = Y^x$ **then**
          **return** 1
      **else**
          **return** 0
      **end if**
  **else**
      **return** 0
  **end if**
```

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

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 $m \in \langle g \rangle$)

ElGamal Security: ElGamal Problems

OW-CPA Security

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

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$
$\mathsf{Gen}(\mathsf{pp}) \xrightarrow{\$} (y, x)$
pick $m$ in group
$\mathsf{Enc}(\mathsf{pp}, y, m) \xrightarrow{\$} (u, v)$
$\mathcal{A}(\mathsf{pp}, y, u, v) \xrightarrow{\$} z$
**return** $1_{z=m}$
```

KR-CPA Security

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

Game

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$
$\mathsf{Gen}(\mathsf{pp}) \xrightarrow{\$} (y, x)$
$\mathcal{A}(\mathsf{pp}, y) \xrightarrow{\$} z$
**return** $1_{x=z}$
```

decryption problem $\iff$ computational Diffie-Hellman problem
key recovery problem $\iff$ discrete logarithm problem

ElGamal KR-CPA Secure $\Longleftrightarrow$ DL Hard

Game DL

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ g, q$
pick $x \in \mathbf{Z}_q$
$X \leftarrow g^x$
$\mathcal{A}(\mathsf{pp}, X) \xrightarrow{\$} z$
return $1_{X=g^z}$
```

Game KR-CPA

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$        $\triangleright\ g, q$
pick $x \in \mathbf{Z}_q$
$y \leftarrow g^x$
$\mathcal{B}(\mathsf{pp}, y) \xrightarrow{\$} z$
return $1_{y=g^z}$
```

$\mathcal{B}(\mathsf{pp}, y)$:

```
$\mathcal{A}(\mathsf{pp}, y) \xrightarrow{\$} z$
return $z$
```

$$\Pr[\text{KR-CPA}(\mathcal{B}) \to 1] = \Pr[\text{DL}(\mathcal{A}) \to 1]$$

ElGamal OW-CPA Secure $\Longrightarrow$ CDH Hard

Game CDH

```
Setup(1^s) $\xrightarrow{\$}$ pp                ▷ $g, q$
pick $x, y \in \mathbf{Z}_q$
$X \leftarrow g^x$, $Y \leftarrow g^y$
$\mathcal{A}(\mathrm{pp}, X, Y) \xrightarrow{\$} K$
return $1_{K=g^{xy}}$
```

Game OW-CPA

```
Setup(1^s) $\xrightarrow{\$}$ pp                ▷ $g, q$
pick $x \in \mathbf{Z}_q$, $y \leftarrow g^x$
pick $m \in \langle g \rangle$
pick $r \in \mathbf{Z}_q$, $u \leftarrow g^r$,
    $v \leftarrow my^r$
$\mathcal{B}(\mathrm{pp}, y, u, v) \xrightarrow{\$} z$
return $1_{m=z}$
```

$\mathcal{B}(\mathrm{pp}, y, u, v)$:

```
7: $\mathcal{A}(\mathrm{pp}, y, u) \xrightarrow{\$} K$
8: return $v/K$
```

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

CDH Hard $\Longrightarrow$ ElGamal OW-CPA Secure

Game OW-CPA

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$    $\triangleright\ g, q$
pick $x \in \mathbf{Z}_q$, $y \leftarrow g^x$
pick $m \in \langle g \rangle$
pick $r \in \mathbf{Z}_q$, $u \leftarrow g^r$,
  $v \leftarrow my^r$
$\mathcal{A}(\mathsf{pp}, y, u, v) \xrightarrow{\$} z$
**return** $1_{m=z}$
```

Game CDH

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} \mathsf{pp}$    $\triangleright\ g, q$
pick $x, y \in \mathbf{Z}_q$
$X \leftarrow g^x$, $Y \leftarrow g^y$
$\mathcal{B}(\mathsf{pp}, X, Y) \xrightarrow{\$} K$
**return** $1_{K=g^{xy}}$
```

```
$\mathcal{B}(\mathsf{pp}, X, Y)$:
  pick $v \in \langle g \rangle$
  $\mathcal{A}(\mathsf{pp}, X, Y, v) \xrightarrow{\$} z$
  **return** $v/z$
```

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

ElGamal Encryption Security

- **key recovery** is equivalent to the discrete logarithm problem
- **decryption** is equivalent to the CDH problem
- **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 m then (u, vw) is the encryption of mw
- OW-CCA attack:

  $\mathcal{A}(\mathrm{pp}, y, u, v)$:

  ```
  1: pick $w$ at random in the group
  2: ODec$(u, vw) \to m'$                ▷ $m' = (vw)/u^x = mw$
  3: **return** $m = m'/w$
  ```

  m 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.*

IND$-CPA 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: $\mathsf{pt} \in \langle g \rangle$

encryption: $\mathsf{Enc}_y(\mathsf{pt}; r) = (g^r, \mathsf{pt}.y^r)$

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

IND-CPA game Γ_0^b:

```
1: run key generation and get $y$
2: run $\mathcal{A}_1(q, g, y) = (\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$
3: pick $r \in_U \mathbf{Z}_q$, and $(u, v) = \mathsf{Enc}_y(\mathsf{pt}_b; r)$
4: run $\mathcal{A}_2(\mathsf{st}, u, v) = b'$
5: return $b'$
```

Transitions — i

```
game $\Gamma_0^b$:
  run key generation and get $y$
  run $\mathcal{A}_1(g, y) = (\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$
  $r \in_U \mathbf{Z}_q$, $(u, v) \leftarrow \mathsf{Enc}_y(\mathsf{pt}_b; r)$
  run $\mathcal{A}_2(\mathsf{st}, u, v) = b'$
  return $b'$
```

$\updownarrow$ bridge

```
game $\Gamma_1^b$:
  pick $x$, $y \leftarrow g^x$
  pick $r$, $u \leftarrow g^r$
  $v_0 \leftarrow g^{xr}$                ▷ erase $x, r$
  run $\mathcal{A}_1(g, y) = (\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$
  $v = \mathsf{pt}_b v_0$
  run $\mathcal{A}_2(\mathsf{st}, u, v) = b'$
  return $b'$
```

$\overset{\text{DDH}}{\approx}$

DDH assumption in the group

```
game $\Gamma_2^b$:
  pick $x$, $y \leftarrow g^x$
  pick $r$, $u \leftarrow g^r$
  pick $v_0$                           ▷ erase $x, r$
  run $\mathcal{A}_1(g, y) = (\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$
  $v = \mathsf{pt}_b v_0$
  run $\mathcal{A}_2(\mathsf{st}, u, v) = b'$
  return $b'$
```

Transitions — ii

```
game $\Gamma_2^b$:
    pick $x$, $y \leftarrow g^x$
    pick $r$, $u \leftarrow g^r$
    pick $v_0$
    run $\mathcal{A}_1(g, y) = (\mathrm{pt}_0, \mathrm{pt}_1, \mathrm{st})$
    $v = \mathrm{pt}_b v_0$
    run $\mathcal{A}_2(\mathrm{st}, u, v) = b'$
    return $b'$
```

$\updownarrow$ bridge

```
game $\Gamma_3^b$:
    pick $x$, $y \leftarrow g^x$
    pick $r$, $u \leftarrow g^r$
    run $\mathcal{A}_1(g, y) = (\mathrm{pt}_0, \mathrm{pt}_1, \mathrm{st})$
    pick $v_0$
    $v = \mathrm{pt}_b v_0$                ▷ erase $v_0$
    run $\mathcal{A}_2(\mathrm{st}, u, v) = b'$
    return $b'$
```

$\overset{\mathrm{ind}}{=}$

messages are in the group!

```
game $\Gamma_4^b$:
    pick $x$, $y \leftarrow g^x$
    pick $r$, $u \leftarrow g^r$
    run $\mathcal{A}_1(g, y) = (\mathrm{pt}_0, \mathrm{pt}_1, \mathrm{st})$
    pick $v$
    run $\mathcal{A}_2(\mathrm{st}, u, v) = b'$
    return $b'$
```

Transitions — iii

final step: Γ_4^0 and Γ_4^1 are identical!

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

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

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: $\langle g \rangle = \mathbf{Z}_n$, $\mathbf{Z}_p^*$, ... ▸ slide 196
- 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 $\text{map}(\text{pt}) = \pm\text{pt} \bmod p \in \langle g \rangle$ 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 $\Longleftrightarrow$ factoring pq
  ElGamal: key recovery $\Longleftrightarrow$ discrete logarithm
- ciphertext **decryption**
  RSA: decryption $\Longleftrightarrow$ RSA problem
  Rabin: decryption $\Longleftrightarrow$ factoring pq
  ElGamal: decryption $\Longleftrightarrow$ CDH problem
- **hard core bit**
  lsb in RSA $\Longleftrightarrow$ RSA problem
- **semantic security** (IND-CPA)
  ElGamal IND-CPA $\Longleftrightarrow$ 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
- security proofs:
  final exam 2010–11 ex4 (security reduction)
  final exam 2011–12 ex2 (MAC revisited)
  final exam 2014–15 ex1 (security of Davies-Meyer)
- DH:
  midterm exam 2010–11 ex2 (easy DDH from bilinear mappings)
  final exam 2011–12 ex1 (easy DDH cases)
  midterm exam 2014–15 ex1 (hard $\log$ and easy DH world)
  final exam 2015–16 ex2 (fixed vs random g in DH problems)
  midterm exam 2016–17 ex1 (solving DDH with small subgroups)
  midterm exam 2017–18 ex2 (gap DH problem)
  midterm exam 2019–20 ex2 (OW-CPA vs IND-CPA)
  midterm exam 2022–23 ex1 (squares in exponent)

Train Yourself

- ElGamal:
  final exam 2009–10 ex2 (message mapping)
  final exam 2012–13 ex1
  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
- MAC vs PRF: midterm exam 2016–17 ex2
- key agreement:
  final exam 2019–20 ex1
  midterm exam 2021–22 ex1
- contact tracing: midterm exam 2019–20 ex2
- equivalent IND notions: midterm exam 2018–19 ex1

Train Yourself

- security of signature with setup: midterm exam 2023–24 ex1
- find-then-guess security for symmetric encryption: midterm exam 2023–24 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

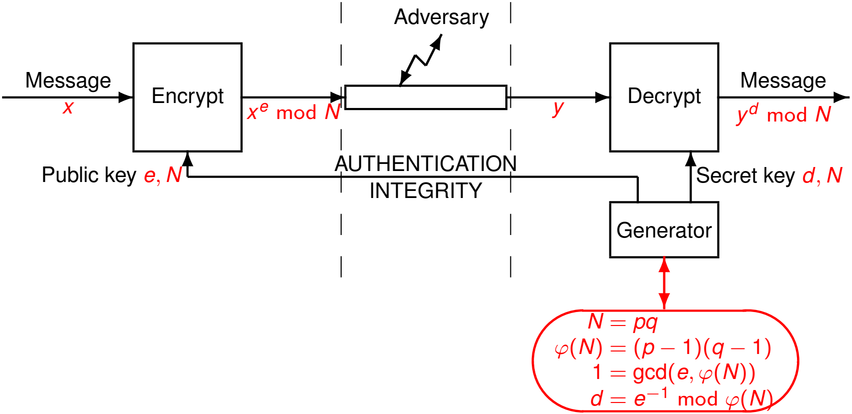
3. Cryptanalysis (Public-Key)
   - RSA
   - Diffie-Hellman
   - ElGamal

4. The Power of Interaction

5. Cryptanalysis (Conventional)

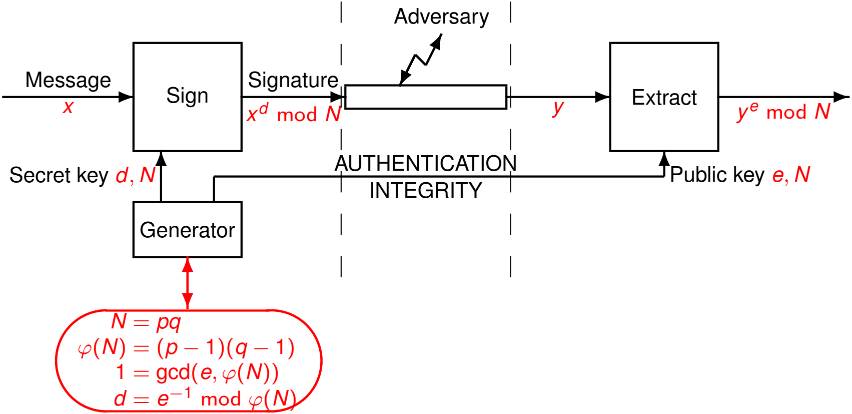
6. Proving Security

Plain RSA Encryption



$$N = pq$$
$$\varphi(N) = (p-1)(q-1)$$
$$1 = \gcd(e, \varphi(N))$$
$$d = e^{-1} \bmod \varphi(N)$$

Plain RSA Signature



$$N = pq$$
$$\varphi(N) = (p-1)(q-1)$$
$$1 = \gcd(e, \varphi(N))$$
$$d = e^{-1} \bmod \varphi(N)$$

RSA Engineering

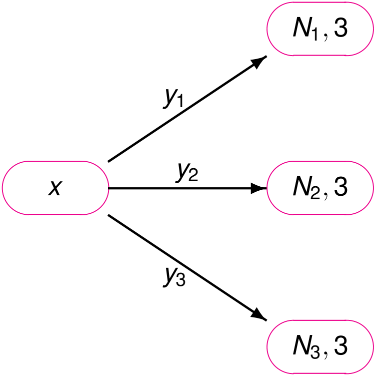
- Implementation issues (from plain RSA to real life standards)
  - Problems with broadcast encryption
  - Problems with low exponents
  - Problems with side channels
- Side channel attacks
  - Power analysis
  - Single/differential fault analysis
  - Timing attack, electromagnetic fields, sound
  - 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 $N_1, \ldots, N_e$.

- The ith participant receives $y_i = x^e \bmod N_i$
- The attacker intercepts e values $y_1, \ldots, y_e$
- The attacker computes $y = x^e \bmod N$ where $N = N_1 \times \ldots \times N_e$ 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 $\mathrm{CRT}(y_1, y_2, y_3) = x^3 \bmod (N_1 N_2 N_3) = x^3$

So we can compute x^3 then extact 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
- Some 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)

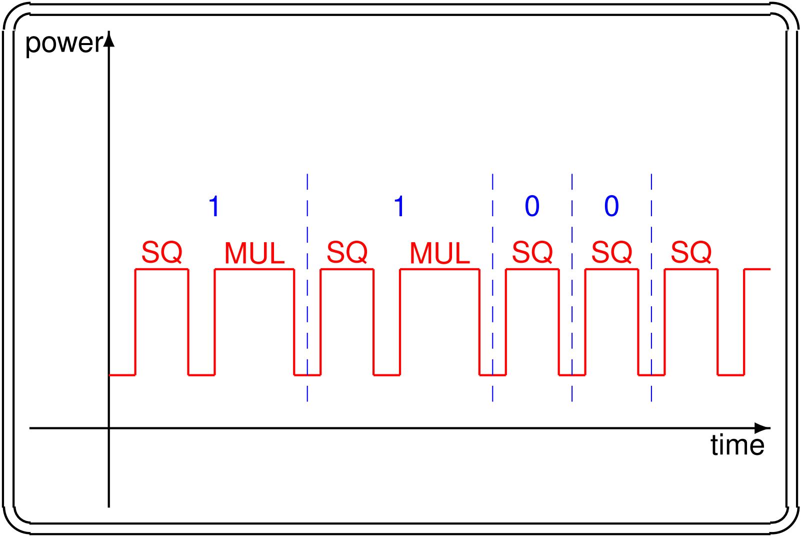
Input: y, d, N, three integers of at most ℓ bits

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

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

```
a ← 1
for $i = \ell - 1$ to 0 do
    $a \leftarrow a^2 \bmod N$
    if $d_i = 1$ then
        $a \leftarrow a \times y \bmod N$
    end if
end for
$x \leftarrow a$
```

SPA



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

Countermeasures

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

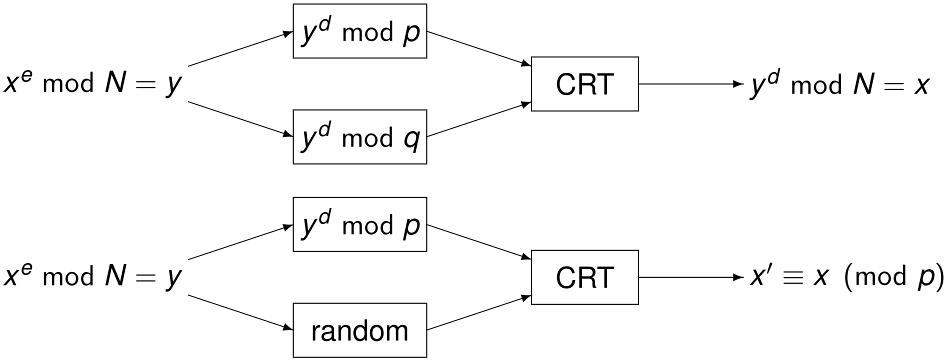
Differential Fault Attack (DFA)

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

- The attacker picks x and sends $y = x^e \bmod N$ to the device
- The attacker agressively (but mildly) stresses the device
- The device eventually makes computation errors
- Error may occur during the CRT accelaration
- 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



$x^e \bmod N = y$

$y^d \bmod p$

$y^d \bmod q$

CRT

$y^d \bmod N = x$

$x^e \bmod N = y$

$y^d \bmod p$

random

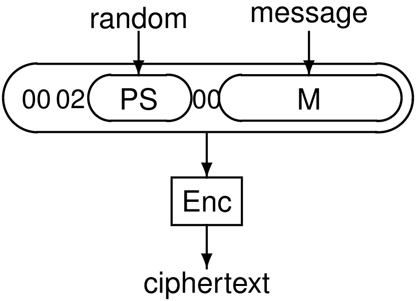
CRT

$x' \equiv x \pmod{p}$

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
- Power LED
- Dynamic frequency scaling (CPU throttling)
- ...

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)

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

Game

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} (\mathsf{pp})$    $\triangleright\ q, g$
pick $x, y \in \mathbf{Z}_q$
$X \leftarrow g^x$, $Y \leftarrow g^y$
$\mathcal{A}(\mathsf{pp}, X, Y) \xrightarrow{\$} K$
return $1_{K = g^{xy}}$
```

DL (Discrete Logarithm)

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

Game

```
$\mathsf{Setup}(1^s) \xrightarrow{\$} (\mathsf{pp})$    $\triangleright\ q, g$
pick $x \in \mathbf{Z}_q$
$X \leftarrow g^x$
$\mathcal{A}(\mathsf{pp}, X) \xrightarrow{\$} z$
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

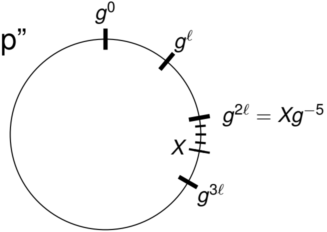
Complexity: $\mathcal{O}(\alpha_1\sqrt{p_1} + \cdots + \alpha_r\sqrt{p_r})$ group operations

```
for $i = 1, \dots, r$ do
    $g' \leftarrow g^{n/p_i^{\alpha_i}}$
    $g'' \leftarrow {g'}^{p_i^{\alpha_i - 1}}$
    $X' \leftarrow X^{n/p_i^{\alpha_i}}$
    $x_i \leftarrow 0$
    for $j = 0$ to $\alpha_i - 1$ do
        $X'' \leftarrow {X'}^{p_i^{\alpha_i - j - 1}}$
        compute the discrete logarithm $u$ of $X''$ in the subgroup of order $p_i$ which is spanned by $g''$ (next slide)
        $X' \leftarrow X'/{g'}^{u.p_i^j}$
        $x_i \leftarrow x_i + u.p_i^j$
    end for
end for
reconstruct and yield $x$ such that $x \equiv x_i \pmod{p_i^{\alpha_i}}$
```

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
  let $\ell = \lceil \sqrt{B} \rceil$ be the size of a “giant step”
  for $i = 0, \ldots, \ell - 1$ do
      set $T\{g^{i\ell}\} \leftarrow i$
  end for
Algorithm
  for $j = 0, \ldots, \ell - 1$ do
      compute $z = Xg^{-j}$
      if $T\{z\}$ exists then
          $i \leftarrow T\{z\}$
          yield $x = i\ell + j$ and stop          ▷ we get $Xg^{-j} = g^{i\ell}$
      end if
  end for
```



Pollard Rho Discrete Logarithm Algorithm

```
Input: $g$ and $X$ in a group $G$ of order $n$
Output: the logarithm of $X$ in base $g$
Complexity: $\mathcal{O}(\sqrt{n})$ group operations
pick a random function $h : G \longrightarrow \{1, 2, 3\}$
$\vec{a}, \vec{b} \leftarrow (1, 0, 0) \in G \times \mathbf{Z}_n \times \mathbf{Z}_n$
repeat
    $\vec{a} \leftarrow f(\vec{a})$
    $\vec{b} \leftarrow f(f(\vec{b}))$
until $a_1 = b_1$
return $(a_2 - b_2)/(a_3 - b_3) \bmod n$    ▷ fail if not possible
```

- G is multiplicatively denoted
- $f(Z, \alpha, \beta)$ is defined to be

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

- vectors (Z, α, β) are all such that $Z = g^\alpha X^\beta$
- we could have taken another random function f with this property

Decisional DH Problem (Reminder)

(implicit: relative to Gen) ▸ slide 144

Hardness of DDHP (Decisional Diffie-Hellman Problem)

$$\mathsf{Adv} = \Pr[\Gamma_1 \to 1] - \Pr[\Gamma_0 \to 1]$$

Game Γ_b

```
Setup($1^s$) $\xrightarrow{\$}$ (pp)                ▷ $q, g$
pick $x, y \in \mathbf{Z}_q$
if $b = 0$ then
    pick $z \in \mathbf{Z}_q$
else
    $z \leftarrow xy \bmod q$
end if
$X \leftarrow g^x$, $Y \leftarrow g^y$, $Z \leftarrow g^z$
$\mathcal{A}(\mathsf{pp}, X, Y, Z) \xrightarrow{\$} c$
return $c$
```

DDH is the key distinguiher problem with DH

Easy Case: $\mathbf{Z}_p^*$

The DDH problem is relative to some Setup may be easy

$\mathbf{Z}_p^*$ with p prime:

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

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

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

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

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

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

Reminder on the Legendre Symbol

- for an odd prime p
- QR_p: set of elements from $\mathbf{Z}_p^*$ which have a square root (quadratic residues)
- $\left(\frac{x}{p}\right) = \begin{cases} 0 & \text{if } x \not\in \mathbf{Z}_p^* \\ +1 & \text{if } x \in \mathrm{QR}_p \\ -1 & \text{if } x \in \mathbf{Z}_p^* - \mathrm{QR}_p \end{cases}$
- $\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \pmod{p}$

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

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

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

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

Hard Cases

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

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

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

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

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

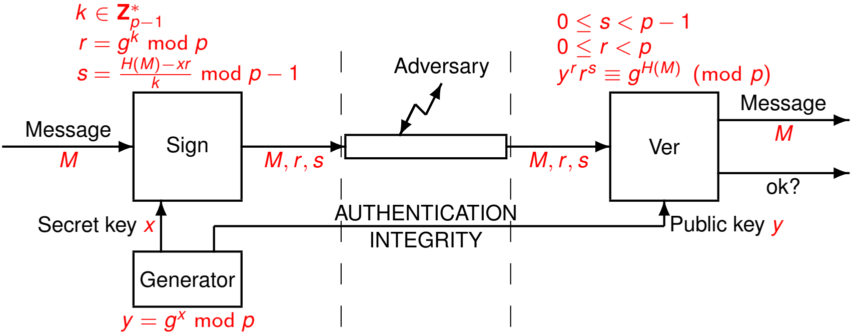
Some Failure Cases

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

ElGamal Signature



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

ElGamal 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 $y^{wc} \bmod p$ in base $g^{wc} \bmod p$ which spans a group of order factor of b. Thus the Pohlig-Hellman algorithm works, thanks to the assumption on b.
- We take $s = t(H(M) - cwz) \bmod (p-1)$.
- 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

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)
  midterm exam 2021–22 ex2 (redundant RSA)
- threshold implementation: midterm exam 2017–18 ex1
- dedicated cryptanalysis: midterm exam 2013–14 ex1
- predicate encryption: midterm exam 2013–14 ex2
- OW-VCA security and Regev: final exam 2020–21 ex1
- Goldwasser-Micali PKC: midterms exam 2018–19 ex2
- a weird signcryption: midterms exam 2018–19 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**
   - Interactive Proofs
   - Zero-Knowledge
   - Zero-Knowledge Construction from Σ 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 $x \in L$

$\mathcal{NP}$ Class

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

co-$\mathcal{NP}$ Class

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 such that for any polynomial P,

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

$\mathcal{P}$ Class

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

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

$=$ language of statements which are checked in polynomial time

NP $\iff$ membership can be decided in **non-deterministic** polynomial time

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

clearly:

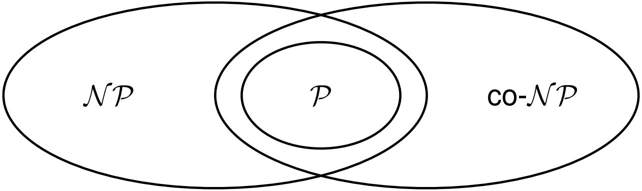
$$\mathcal{P} \subseteq \mathcal{NP}$$

big open question:

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

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

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

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

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

$$\text{SAT} \in \mathcal{P} \quad \Longleftrightarrow \quad \mathcal{P} = \mathcal{NP}$$

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 $m_1, \ldots, m_n$ of **incoming messages** to an **outgoing message** $\mathcal{A}(x, m_1, \ldots, m_n; r)$.

- 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 $\mathcal{A}(x, m_1, \ldots, m_n; r)$ is a terminal message.
- $(x, m_1, \ldots, m_n; r)$ 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

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

$$\begin{aligned} a_i &= \mathcal{A}(x, b_1, \ldots, b_{i-1}; r_A) \\ b_j &= \mathcal{B}(x, a_1, \ldots, a_j; r_B) \end{aligned}$$

  for $i = 1, \ldots, n_A$ s.t. a_{n_A} is the first terminal message a_i and $j = 1, \ldots, n_B$ s.t. b_{n_B} is the first terminal message b_j
- $\mathrm{Out}_{\mathcal{A}}(\exp) = a_{n_A}$
- $\mathrm{Out}_{\mathcal{B}}(\exp) = b_{n_B}$
- $\mathrm{View}_{\mathcal{A}}(\exp) = (x, b_1, \ldots, b_{n_A - 1}; r_A)$
- $\mathrm{View}_{\mathcal{B}}(\exp) = (x, a_1, \ldots, a_{n_B}; r_B)$

Interactive System

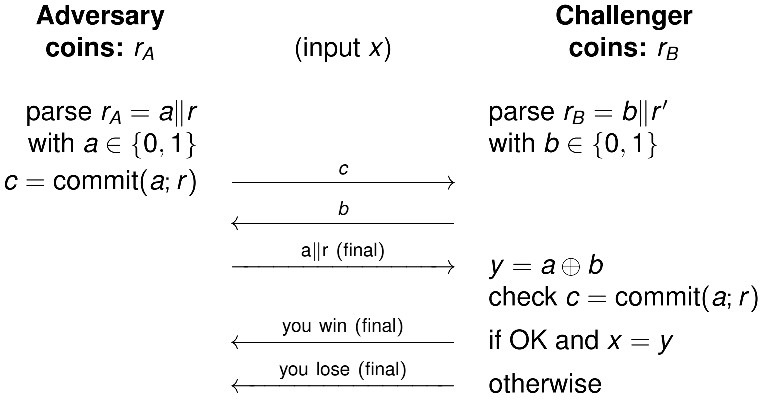
$$
\begin{array}{rcl}
\mathcal{A} & & \mathcal{B} \\
\textbf{coins: } r_A & \text{(input } x\text{)} & \textbf{coins: } r_B \\
a_1 = \mathcal{A}(x; r_A) & \xrightarrow{\qquad a_1 \qquad} & \\
 & \xleftarrow{\qquad b_1 \qquad} & b_1 = \mathcal{B}(x, a_1; r_B) \\
a_2 = \mathcal{A}(x, b_1; r_A) & \xrightarrow{\qquad a_2 \qquad} & \\
 & \vdots & \\
\mathcal{A}(x, b_1, \ldots, b_{n_A - 1}; r_A) & \xrightarrow{\qquad a_{n_A} \qquad} & \\
 & \xleftarrow{\qquad b_{n_B} \qquad} & \mathcal{B}(x, a_1, \ldots, a_{n_B}; r_B)
\end{array}
$$

$$
\begin{array}{rcl}
\text{View}_{\mathcal{A}}(\text{exp}) & = & (x, b_1, \ldots, b_{n_A - 1}; r_A) \\
\text{View}_{\mathcal{B}}(\text{exp}) & = & (x, a_1, a_2, \ldots, a_{n_B}; r_B)
\end{array}
$$

Example: Coin Flipping Game

- **game**:
  given a bit x, $\mathcal{A}$ and $\mathcal{C}$ flip a coin together and $\mathcal{A}$ wins if it is x
- **coin flipping**:
  1. $\mathcal{A}$ commits to a random bit a
  2. $\mathcal{C}$ flips a bit b and sends it
  3. $\mathcal{A}$ opens a
  4. the final bit is $a \oplus b$

Example: Coin Flipping Game



Adversary
coins: r_A

(input x)

Challenger
coins: r_B

parse $r_A = a\|r$
with $a \in \{0,1\}$

parse $r_B = b\|r'$
with $b \in \{0,1\}$

$c = \text{commit}(a;r)$ $\xrightarrow{\quad c \quad}$

$\xleftarrow{\quad b \quad}$

$\xrightarrow{\text{a}\|\text{r (final)}}$ $y = a \oplus b$

check $c = \text{commit}(a;r)$

$\xleftarrow{\text{you win (final)}}$ if OK and $x = y$

$\xleftarrow{\text{you lose (final)}}$ otherwise

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

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

Interactive Proof

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, β such that $0 \leq \beta < 1$ and

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

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

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

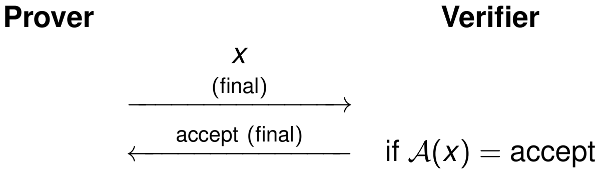
$$\Pr_{r_P, r_V}\left[\mathsf{Out}_{\mathcal{V}}\left(\mathcal{P}^*(r_P) \overset{x}{\leftrightarrow} \mathcal{V}(r_V)\right) = \text{accept}\right] \leq \beta$$

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
- β is the maximal probability that $\mathcal{V}$ can be fooled

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

language: set of x such that $\mathcal{A}(x) = \text{accept}$ where $\mathcal{A}$ is a deterministic and polynomially bounded Turing machine



Prover

Verifier

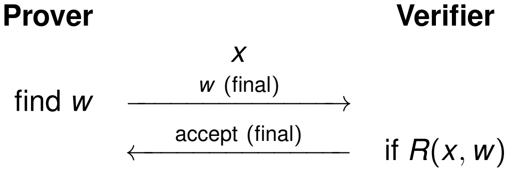
x (final) →

← accept (final)

if $\mathcal{A}(x) = \text{accept}$

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

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



Prover

Verifier

x

find w $\xrightarrow{w \text{ (final)}}$

$\xleftarrow{\text{accept (final)}}$ if $R(x,w)$

Example: Goldwasser-Micali-Rackoff 1985

A co-NP Case

reminder: QR(n) is the set of all squares modulo n

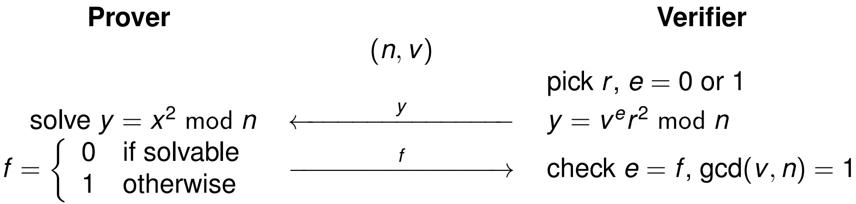
language: set of pairs (n, v) such that $v \in \mathbf{Z}_n^*$ and $v \notin \mathrm{QR}(n)$

$$
\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
 & (n, v) & \\
 & & \text{pick } r \in \mathbf{Z}_n^*,\ e \in \{0, 1\} \\
\text{solve } y = x^2 \bmod n & \xleftarrow{\hspace{3em} y \hspace{3em}} & y = v^e r^2 \bmod n \\
f = \begin{cases} 0 & \text{if solvable} \\ 1 & \text{otherwise} \end{cases} & \xrightarrow{\hspace{2em} f \ \text{(final)} \hspace{2em}} & \\
 & \xleftarrow{\hspace{1em} \text{accept (final)} \hspace{1em}} & \text{if } e = f \text{ and } \gcd(v, n) = 1
\end{array}
$$

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

Example: GMR85 Completeness

completeness:



Prover

Verifier

(n, v)

pick r, $e = 0$ or 1

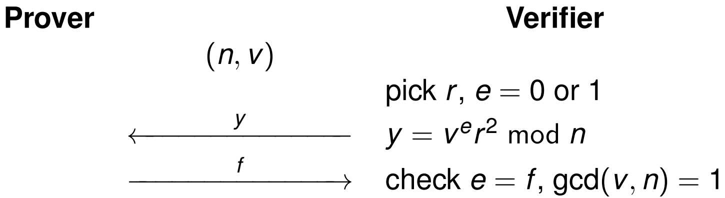
solve $y = x^2 \bmod n$ $\xleftarrow{y}$ $y = v^e r^2 \bmod n$

$f = \begin{cases} 0 & \text{if solvable} \\ 1 & \text{otherwise} \end{cases}$ $\xrightarrow{f}$ check $e = f$, $\gcd(v, n) = 1$

- 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

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



Prover

Verifier

(n, v)

pick r, $e = 0$ or 1

$\xleftarrow{\quad y \quad}$ $y = v^e r^2 \bmod n$

$\xrightarrow{\quad f \quad}$ check $e = f$, $\gcd(v, n) = 1$

- assume that $\mathcal{V}$ follows the protocol and $(n, v) \not\in L$
- 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 β-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 β'-sound with

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

so $\beta' \to 0$

Conclusion: by **sequential** composition we can tune β as close to 0 as we want

Soundness Result for Sequential Composition

Theorem

Consider an interactive proof system $(\mathcal{P}, \mathcal{V})$ which is β-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 β'-sound where*

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

▸ skip details

Technical Lemma

Lemma

Assume that no adversary can succeed one protocol session with probability higher than β.
Then, for each adversary $\mathcal{A}$ repeating the protocol n times and any $I \subseteq \{1, \ldots, n\}$, we have

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

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

Proof of Lemma

- proceed by induction on i such that $I \subseteq \{1, \ldots, i\}$:
  - trivial for $i = 0$ (I empty)
  - assume this is true for $i - 1$ and prove it for i:
    - let $I \subseteq \{1, \ldots, i\}$
    - if $I \subseteq \{1, \ldots, i-1\}$, 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 ith iteration
    - if $\mathcal{B}$ halts, since he passes with probability at most β, we have
      $$\Pr\left[\mathcal{A}\text{ succeeds iteration } i \,\middle|\, \bigwedge_{j \in I'} \mathcal{A}\text{ succeeds iteration } j\right] \leq \beta$$
    - by induction, we prove (if $\mathcal{B}$ does not halt as well) that
      $$\Pr\left[\bigwedge_{i \in I} \mathcal{A}\text{ succeeds iteration } i\right] \leq \beta^{\#I}$$

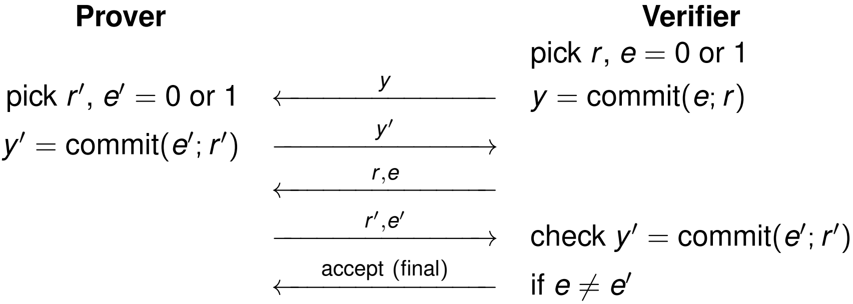
□

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



Prover **Verifier**

pick r, $e = 0$ or 1

pick r', $e' = 0$ or 1 $\xleftarrow{\quad y \quad}$ $y = \text{commit}(e; r)$

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

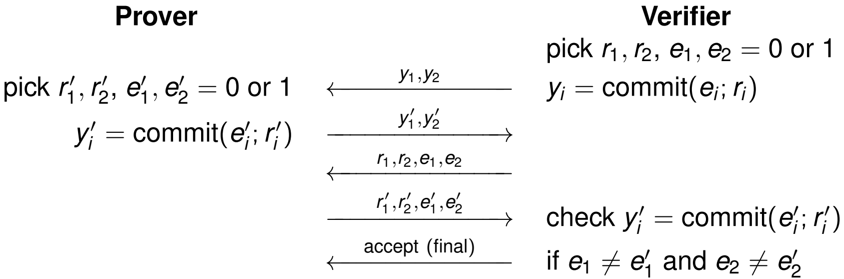
$\xleftarrow{\quad r, e \quad}$

$\xrightarrow{\quad r', e' \quad}$ check $y' = \text{commit}(e'; r')$

$\xleftarrow{\text{accept (final)}}$ if $e \neq e'$

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

Parallel Composition of DD

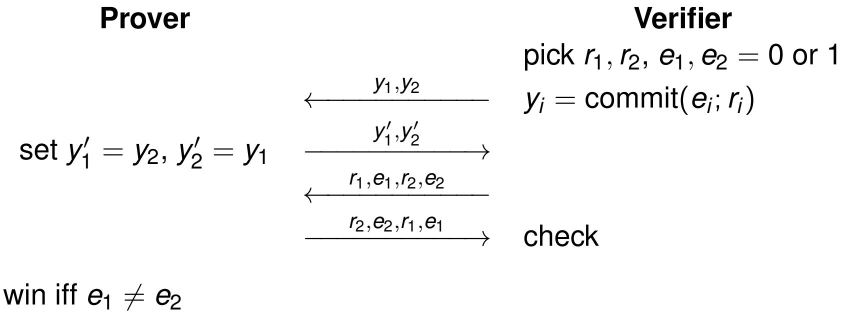


Prover		**Verifier**
		pick $r_1, r_2, e_1, e_2 = 0$ or 1
pick $r'_1, r'_2, e'_1, e'_2 = 0$ or 1	$\xleftarrow{y_1, y_2}$	$y_i = \text{commit}(e_i; r_i)$
$y'_i = \text{commit}(e'_i; r'_i)$	$\xrightarrow{y'_1, y'_2}$	
	$\xleftarrow{r_1, r_2, e_1, e_2}$	
	$\xrightarrow{r'_1, r'_2, e'_1, e'_2}$	check $y'_i = \text{commit}(e'_i; r'_i)$
	$\xleftarrow{\text{accept (final)}}$	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 $\frac{1}{4}$)



Prover

Verifier

pick $r_1, r_2, e_1, e_2 = 0$ or 1

$\xleftarrow{y_1,y_2}$ $y_i = \text{commit}(e_i; r_i)$

set $y'_1 = y_2$, $y'_2 = y_1$ $\xrightarrow{y'_1,y'_2}$

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

$\xrightarrow{r_2,e_2,r_1,e_1}$ check

win iff $e_1 \neq e_2$

$\mathcal{IP}$ Class

Definition ($\mathcal{IP}$ Language)

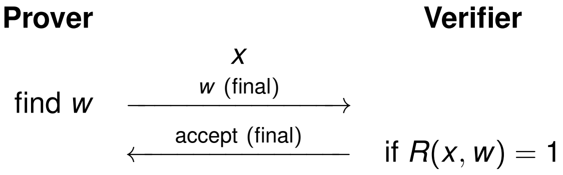
A language L over the alphabet Z belongs to the class $\mathcal{IP}$ if there exists an interactive proof system for L.

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.



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

$\mathcal{PSPACE}$ Class

Definition ($\mathcal{PSPACE}$ Language)

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

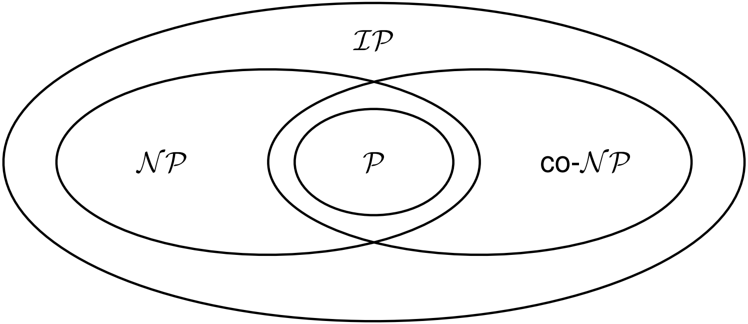
$$L = \{x \in Z^*; \mathcal{A}(x) \rightarrow \text{accept}\}$$

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

Theorem (Shamir 1992)

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

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 Σ 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 $x \in L$ $\text{View}_{\mathcal{V}^*}\left(\mathcal{P}(r_P) \overset{x}{\leftrightarrow} \mathcal{V}^*(r_V)\right)$ and $\mathcal{S}(x; r)$ produce $*$-indistinguishable distributions.

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

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 $|x|$

Zero-Knowledge Levels

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

$\mathsf{View}_{\mathcal{V}^*}(x)$: what comes from interacting with powerful prover
$\mathsf{Out}_{\mathcal{S}}(x)$: what comes from from powerless simulator

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

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

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

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

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

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

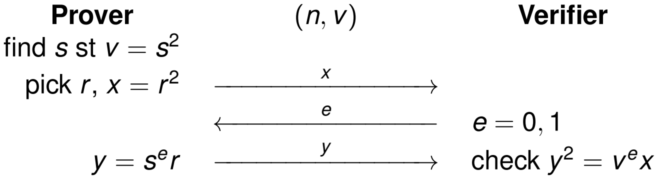
Example: Goldwasser-Micali-Rackoff 1989

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

Prover		**Verifier**
	(n, v)	
find s st $v = s^2 \bmod n$		
pick $r \in \mathbf{Z}_n^*$, $x = r^2 \bmod n$	$\xrightarrow{\quad x \quad}$	
	$\xleftarrow{\quad e \quad}$	$e = 0$ or 1
$y = s^e r \bmod n$	$\xrightarrow{\quad y \quad}$	check $y^2 \equiv v^e x \pmod{n}$ and $\gcd(x, n) = 1$

GMR89 - Completeness

completeness:



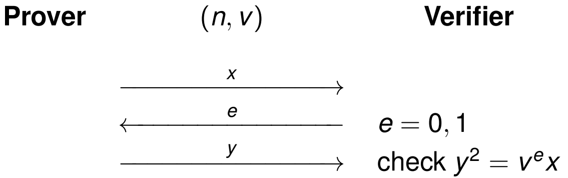
$$\begin{array}{rcl}
\textbf{Prover} & (n, v) & \textbf{Verifier} \\
\text{find } s \text{ st } v = s^2 & & \\
\text{pick } r,\, x = r^2 & \xrightarrow{\qquad\qquad x \qquad\qquad} & \\
 & \xleftarrow{\qquad\qquad e \qquad\qquad} & e = 0, 1 \\
y = s^e r & \xrightarrow{\qquad\qquad y \qquad\qquad} & \text{check } y^2 = v^e x
\end{array}$$

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

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

GMR89 - Soundness

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



Prover (n, v) **Verifier**

$\xrightarrow{x}$

$\xleftarrow{e}$ $e = 0, 1$

$\xrightarrow{y}$ check $y^2 = v^e x$

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

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

- 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

$$\begin{array}{rcl}
\textbf{Prover} & (n, v) & \textbf{Verifier} \\
\text{find } s \text{ st } v = s^2 \bmod n & & \\
\text{pick } r,\ x = r^2 & \xrightarrow{\quad\quad\quad x \quad\quad\quad} & \\
 & \xleftarrow{\quad\quad\quad e \quad\quad\quad} & \\
y = s^e r & \xrightarrow{\quad\quad\quad y \quad\quad\quad} &
\end{array}$$

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 $(n, v) \in L$,

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

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

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

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

regular ZK:	$\forall\mathcal{V}^* \; \exists\mathcal{S} \; \forall x$	$\text{View}_{\mathcal{V}^*}(x) \sim \text{Out}_{\mathcal{S}}(x)$	▸ slide 255
BB ZK:	$\exists\mathcal{S} \; \forall\mathcal{V}^* \; \forall x$	$\text{View}_{\mathcal{V}^*}(x) \sim \text{Out}_{\mathcal{S}}(x)$	

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 $\varphi : V \to \{1, 2, 3\}$ such that for each $(u, v) \in E$ then $\varphi(u) \neq \varphi(v)$

Prover		**Verifier**
	(V, E)	
find φ		
	repeat $\#E$ times	
pick $\pi \in S_3$		
r_u for each $u \in V$		
$c_u = \pi(\varphi(u))$		
$R_u = \text{commit}(c_u, r_u)$	$\xrightarrow{R}$	
	$\xleftarrow{u,v}$	pick $(u, v) \in E$
if $(u, v) \in E$	$\xrightarrow{c_u,c_v,r_u,r_v}$	check R_u, R_v
		check $c_u \neq c_v$

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 $\varphi : V \to \{1, 2, \ldots, n\}$ 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:

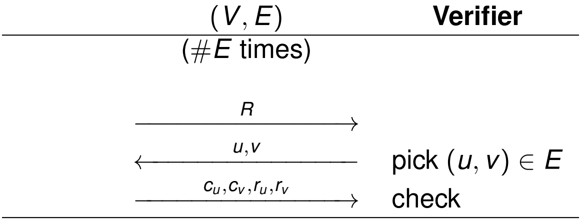
Prover φ	(V, E)	**Verifier**
pick $\pi \in S_3$, r	$(\#E$ times$)$	
$c_u = \pi(\varphi(u))$		
$R_u = \text{commit}(c_u, r_u)$	$\xrightarrow{R}$	
	$\xleftarrow{u,v}$	pick $(u, v) \in E$
	$\xrightarrow{c_u,c_v,r_u,r_v}$	check

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

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

GMW86 - Soundness (sketch)

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



(V, E) **Verifier**

$(\#E$ times$)$

$\xrightarrow{R}$

$\xleftarrow{u,v}$ pick $(u, v) \in E$

$\xrightarrow{c_u,c_v,r_u,r_v}$ check

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

- if there is no possible φ 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)

Prover φ		(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

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

- 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 $L \in \mathcal{NP}$ *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 β such that $0 \leq \beta < 1$ and

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

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

  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:

1. $\mathcal{P}$ finds w such that $R(x, w)$ by exhaustive search
2. $\mathcal{P}$ 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 Σ Protocol
   - Setup Models
   - A Building Block for Making Cryptographic Primitives

5. Cryptanalysis (Conventional)

6. Proving Security

Motivation

Σ-protocols make the design of ZK proofs of knowledge easier

Σ Protocol

Definition

Given a language $L \in \mathcal{NP}$ over an alphabet Z defined by a relation R, a Σ-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 $\mathcal{P}(r_P) \overset{x}{\leftrightarrow} \mathcal{V}(r_V)$ conditioned to e has same distribution as $\mathcal{S}(x, e; r)$. $(\Pr_{\mathcal{P}\leftrightarrow\mathcal{V}}[a, z | x, e] = \Pr_{\mathcal{S}}[a, z | x, e])$

Σ Protocol

$$
\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
w & x & \\
a \leftarrow \mathcal{P}(x, w; r_P) & \xrightarrow{\qquad\qquad a \qquad\qquad} & \\
 & \xleftarrow{\qquad\qquad e \qquad\qquad} & \text{pick } e \in_U E \\
z \leftarrow \mathcal{P}(x, w, e; r_P) & \xrightarrow{\qquad\qquad z \qquad\qquad} & \text{check } V(x, a, e, z)
\end{array}
$$

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

Specifying a Σ Protocol

To fully define a Σ-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. $\forall x \; \forall e, e' \in E \; \forall a, z, z'$ $(e \neq e', V(x, a, e, z), V(x, a, e', z')) \Longrightarrow R(x, \mathcal{E}(x, a, e, z, e', z'))$
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 $R((G_0, G_1), \varphi)$: φ invertible and $\varphi(G_0) = G_1$
- $\mathcal{P}(G_0, G_1, \varphi; r_P)$, e domain, $\mathcal{P}(G_0, G_1, \varphi, e; r_P)$, $V(G_0, G_1, H, e, \sigma)$

Prover		**Verifier**
φ st $\varphi(G_0) = G_1$	(G_0, G_1)	
pick π invertible		pick $e \in \{0, 1\}$
$H = \pi(G_0)$	$\xrightarrow{\quad H \quad}$	
	$\xleftarrow{\quad e \quad}$	
$\sigma = \pi \circ \varphi^{-e}$	$\xrightarrow{\quad \sigma \quad}$	$\sigma(G_e) \overset{?}{=} H$

$$\begin{tikzcd}
G_0 \arrow[rr, bend left, "\varphi"] \arrow[dr, "\pi"] & & G_1 \arrow[dl, "\pi\circ\varphi^{-1}"] \\
& H &
\end{tikzcd}$$

- $\mathcal{E}(G_0, G_1, H, e, \sigma_e, e', \sigma_{e'}) = \sigma_1^{-1} \circ \sigma_0$ ($e \neq e'$ so $\{e, e'\} = \{0, 1\}$)
- $\mathcal{S}(G_0, G_1, e; r)$: pick σ 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 **egdes**
  an edge $(u, v) \in E$ is said to go from vertex u to vertex v
- given a permutation π 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 π over V such that $\pi(G) = G'$

Check List

1. R, $\mathcal{P}$, V, $\mathcal{E}$, $\mathcal{S}$ and sampling are polynomially computable in $|G_0, G_1|$
2. perfect completeness: quite clear
3. $\forall(G_0, G_1, \varphi) \in R \; \forall H, \sigma_0, \sigma_1$ $(V(G_0, G_1, H, 0, \sigma_0), V(G_0, G_1, H, 1, \sigma_1)) \Longrightarrow$ $R(G_0, G_1, \mathcal{E}(G_0, G_1, H, 0, \sigma_0, 1, \sigma_1))$
4. $\forall(G_0, G_1, \varphi) \in R \; \forall e \quad \text{distrib}_{r_P}(H, e, \sigma) =$ $\text{distrib}(\mathcal{S}(G_0, G_1, e))$
    - since H is a function of G_0, G_1, e, σ, 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 for V of form

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

(most of Σ-protocols are like this)

- define

  $\mathcal{S}(x, e; r)$:
  ```
  1: pick $z$
  2: set $a = f(x, e, z)$
  ```
- use z well distributed, i.e. such that:

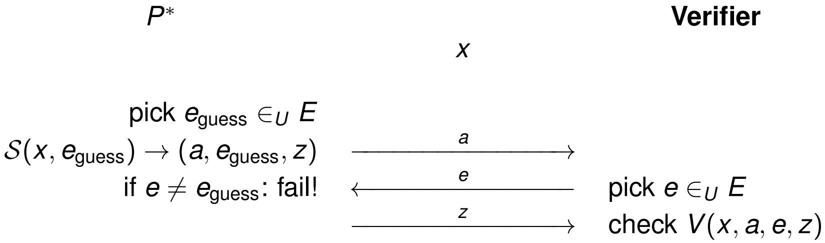
$$\forall x, e, z \, \Pr[\mathrm{View}_{\mathcal{V}} \to z | x, e] = \Pr[\mathcal{S} \to z | x, e]$$

- conclude:

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

A Malicious Prover

$\mathcal{S}$: as on previous slide



P^* **Verifier**

x

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

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

if $e \neq e_{\text{guess}}$: fail! $\xleftarrow{\quad e \quad}$ pick $e \in_U E$

$\xrightarrow{\quad z \quad}$ check $V(x, a, e, z)$

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

Σ Protocols are Proof of Knowledge — i

Theorem

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

Proof. Termination and completeness come from the definition. Extractor (sketch):

- pick r_P, r_V, r'_V and run $\mathcal{P}^*(r_P) \overset{x}{\leftrightarrow} \mathcal{V}(r_V)$ and $\mathcal{P}^*(r_P) \overset{x}{\leftrightarrow} \mathcal{V}(r'_V)$
- we have same a, if $e \neq e'$ and the two runs accept then $\mathcal{E}(x, a, e, z, e', z')$ yields w
- 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

Σ Protocols are Proof of Knowledge — ii

run 1

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

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

run 2

$\mathcal{P}^*(r_P) \overset{x}{\leftrightarrow} \mathcal{V}(r'_V)$ 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

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

Σ Protocols are Proof of Knowledge — iii

- $\Pr[bb' = 1, e = e' | r_P] = \varepsilon(x, r_P)\beta$
- $\Pr[bb' = 1, e \neq e' | r_P] = \varepsilon(x, r_P)(\varepsilon(x, r_P) - \beta)$
- 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 $\mathcal{O}\left(\frac{\mathrm{Poly}(|x|)}{\varepsilon(x)-\beta}\right)$ □

Parallel Composition

Theorem

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

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

$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
w \text{ st } R(x,w) & x & \\
\text{pick } r_1,\ldots,r_t & & \text{pick } (e_1,\ldots,e_t)\in E^t \\
a_i=\mathcal{P}(x,w;r_i) & \xrightarrow{a_1,\ldots,a_t} & \\
 & \xleftarrow{e_1,\ldots,e_t} & \\
z_i=\mathcal{P}(x,w,e_i;r_i) & \xrightarrow{z_1,\ldots,z_t} & \bigwedge_i V(x,a_i,e_i,z_i)?
\end{array}$$

- $\mathcal{E}(x,a,e,z,e',z')$:
  find i s.t. $e_i\neq e'_i$ then do $\mathcal{E}_{\text{old}}(x,a_i,e_i,z_i,e'_i,z'_i)$
- $\mathcal{S}(x,e_1,\ldots,e_t;r_1,\ldots,r_t)$:
  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

$$\mathrm{View}_{\mathcal{V}}\left(\mathcal{P}(r_P) \overset{x}{\leftrightarrow} \mathcal{V}(r_V)\right)$$

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

“ZK only when the verifier is honest”

Σ Protocols are HVZK

Theorem

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

Proof. We construct a simulator for the honest $\mathcal{V}$ as follows:

```
1: pick $a_0$, set $e = \mathcal{V}(x, a_0; r_V)$            ▷ $\mathcal{V}(.,.; r_V)$ is constant
2: pick $r$, compute $(a, e, z) = \mathcal{S}(x, e; r)$
3: yield $(x, a, z; r_V)$ as the view                         ▷ note: $\mathcal{V}(x, a; r_V) = e$
```

- $\Pr[x, a, z, r_V] = \Pr[x, a, z | r_V] \Pr[r_V]$ (Bayes)
- $\Pr[x, a, z | r_V] = \Pr[x, a, z | e, r_V]$ (e is a function of r_V)
- $\Pr[x, a, z | e, r_V] = \Pr[x, a, z | e]$ (only e depends on r_V)
- $\Pr[\text{view}] = \Pr[x, a, z | e] \Pr[r_V] = \Pr[a, z | x, e] \Pr[x] \Pr[r_V]$
- $\Pr_{\mathcal{P} \leftrightarrow \mathcal{V}}[a, z | x, e] = \Pr_{\mathcal{S}}[a, z | x, e]$ due to special HVZK

□

Zero-Knowledge on Small Challenge Set

Theorem

A Σ-protocol with a challenge set E with polynomially bounded size is zero-knowledge.

Proof. Simulator:

```
pick $e_{\text{guess}} \in E$                                   ▷ a guess for $e$
run $\mathcal{S}(x, e_{\text{guess}}) \rightarrow (a, e_{\text{guess}}, z)$
get $e^* = \mathcal{V}^*(a; \rho)$ for $\rho$ random
if $e^* \neq e_{\text{guess}}$: try again                        ▷ this trial failed
output $(a, z; \rho)$
```

- (a, e_{guess}, z) has same distribution as (a, e, z) in $\mathcal{P} \overset{x}{\leftrightarrow} \mathcal{V}$
- same as

```
run $\mathcal{P} \overset{x}{\leftrightarrow} \mathcal{V}$ where $\mathcal{V}$ selects a random $e$
if $e \neq \mathcal{V}^*(a; \rho)$ for $\rho$ random: try again
output $(a, z; \rho)$
```

- $\Pr[\neg\text{try again}] = 1/\#E$ so it terminates with polynomial time
- final $(a, z; \rho)$ has same distribution as for $\mathcal{P} \overset{x}{\leftrightarrow} \mathcal{V}^*$

Σ Protocols are Not Always ZK

For a malicious $\mathcal{V}^*$ who sets $e = H(x, a)$:

- assume that e is large ($\#E$ super-polynomial)
- assume that H “looks like random” (to be formalized later)
- assume that finding a witness for x is hard
- then simulating the proof with $\mathcal{V}^*$ is hard
- $\rightarrow$ not ZK
- (Fiat-Shamir result to be seen later)

Summary about Composition

	parallel composition	**sequential composition**
proof systems	it works	it works
Σ-protocols	it works	*it does not work*
ZK proofs	it may not work	it works

- proof systems: soundness amplifies well
- Σ-protocols: a sequential repetition is no longer a Σ-protocol
- ZK proofs: counterexample

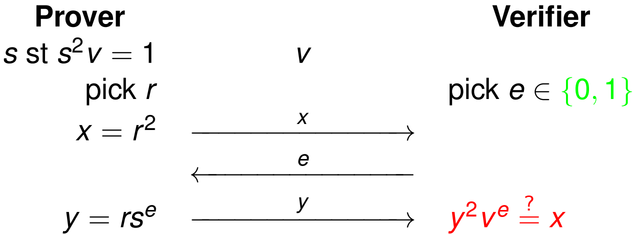
  Σ-protocols are fully ZK if E is small (polynomial size) but not fully ZK if E is large!

we don't obtain a Σ-protocol

Example: Fiat-Shamir 1986 Simplified

$\approx$GMR89 ▸ slide 257

- relation $R(v, s)$: $s^2 v = 1$
- $\mathcal{P}(v, s; r_P)$, domain for e, $\mathcal{P}(v, s, e; r_P)$, $V(v, x, e, y)$



$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
s \text{ st } s^2 v = 1 & v & \\
\text{pick } r & & \text{pick } e \in \{0, 1\} \\
x = r^2 & \xrightarrow{\quad x \quad} & \\
 & \xleftarrow{\quad e \quad} & \\
y = r s^e & \xrightarrow{\quad y \quad} & y^2 v^e \stackrel{?}{=} x
\end{array}$$

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

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

- $\mathcal{S}(v, e)$: pick y then set $x = y^2 v^e$

Example: Fiat-Shamir 1986

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

$$
\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
s \text{ st } s^2v \bmod n = 1 & (n,v) & \\
\text{pick } r \in \mathbf{Z}_n^* & & \text{pick } e \in \{0,1\} \\
x = r^2 \bmod n & \xrightarrow{\qquad x \qquad} & \\
 & \xleftarrow{\qquad e \qquad} & \\
y = rs^e \bmod n & \xrightarrow{\qquad y \qquad} & y^2v^e \bmod n \overset{?}{=} x \\
 & & v, y \overset{?}{\in} \mathbf{Z}_n^*
\end{array}
$$

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

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

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

Check List

1. R, $\mathcal{P}$, V, $\mathcal{E}$, $\mathcal{S}$ and sampling are polynomially computable in $|\text{instance}|$
2. perfect completeness: quite clear
3. special soundness: in previous slide
4. $\forall(\text{instance}, s) \in R\ \forall e \quad \text{distrib}_{r_P}(x, e, y) = \text{distrib}(\mathcal{S}(\text{instance}, e))$
   - since x is a function of instance, e, y, it is enough to show that $\forall(\text{instance}, s) \in R\ \forall e$, $y = rs^e \bmod n$ 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 Simplified

- relation $R(y, x)$: $g^x = y$ in a group $\langle g \rangle$
- $\mathcal{P}(y, x; r_P)$, domain for e, $\mathcal{P}(y, x, e; r_P)$, $V(y, r, e, s)$

$$
\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
x \text{ st } g^x = y & y & \\
\text{pick } k & & \text{pick } e \text{ (can be large)} \\
r = g^k & \xrightarrow{\qquad\qquad r \qquad\qquad} & \\
 & \xleftarrow{\qquad\qquad e \qquad\qquad} & \\
s = ex + k & \xrightarrow{\qquad\qquad s \qquad\qquad} & ry^e \stackrel{?}{=} g^s
\end{array}
$$

- $\mathcal{E}(y, r, e, s, e', s') = \frac{s-s'}{e-e'}$

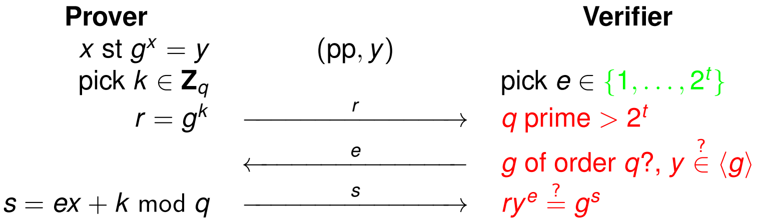
$$g^{s-s'} = \frac{ry^e}{ry^{e'}} = y^{e-e'}$$

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

- $\mathcal{S}(y, e)$: pick s then set $r = g^s y^{-e}$

Example: Schnorr 1989

- pp defines g, q, and a group $\langle g \rangle$ of prime order q
- relation $R((\text{pp}, y), x)$: $g^x = y$, q prime $> 2^t$, g of order q
- $\mathcal{P}(\text{pp}, y, x; r_P)$, domain for e, $\mathcal{P}(\text{pp}, y, x, e; r_P)$, $V(\text{pp}, y, r, e, s)$



Prover **Verifier**

x st $g^x = y$ (pp, y)

pick $k \in \mathbf{Z}_q$ pick $e \in \{1, \ldots, 2^t\}$

$r = g^k$ $\xrightarrow{r}$ q prime $> 2^t$

$\xleftarrow{e}$ g of order q?, $y \overset{?}{\in} \langle g \rangle$

$s = ex + k \bmod q$ $\xrightarrow{s}$ $ry^e \overset{?}{=} g^s$

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

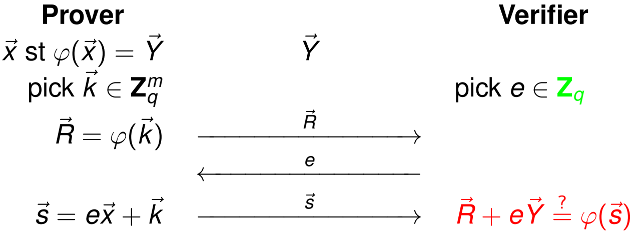
$$g^{s-s'} = \frac{ry^e}{ry^{e'}} = y^{e-e'}$$

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

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

Example: Schnorr 1989 Generalized

- group homomorphism $\varphi : \mathbf{Z}_q^m \to G^n$, prime q
- relation $R((\varphi, \vec{Y}), \vec{x})$: $\varphi(\vec{x}) = \vec{Y}$
- $\mathcal{P}(\vec{Y}, \vec{x}; r_P)$, domain for e, $\mathcal{P}(\vec{Y}, \vec{x}, e; r_P)$, $V(\vec{Y}, \vec{R}, e, \vec{s})$



Prover **Verifier**

$\vec{x}$ st $\varphi(\vec{x}) = \vec{Y}$ $\vec{Y}$

pick $\vec{k} \in \mathbf{Z}_q^m$ pick $e \in \mathbf{Z}_q$

$\vec{R} = \varphi(\vec{k})$ $\xrightarrow{\vec{R}}$

$\xleftarrow{e}$

$\vec{s} = e\vec{x} + \vec{k}$ $\xrightarrow{\vec{s}}$ $\vec{R} + e\vec{Y} \overset{?}{=} \varphi(\vec{s})$

- example: proof of discrete log equality $g_1^x = y_1$ and $g_2^x = y_2$
- example: proof of representation $g_1^{x_1} g_2^{x_2} g_3^{x_3} = y$

Check List

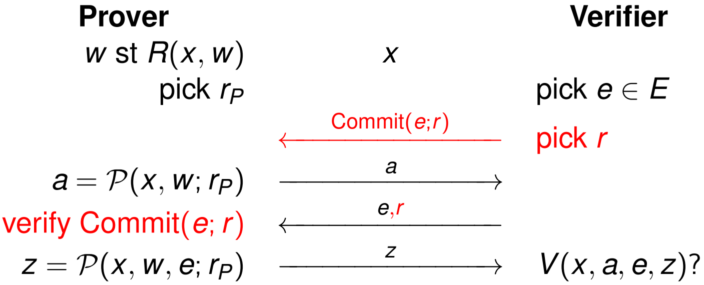
1. R, $\mathcal{P}$, V, $\mathcal{E}$, $\mathcal{S}$ and sampling are polynomially computable in $|\text{instance}|$
2. perfect completeness: quite clear
3. special soundness: in previous slide
4. $\forall(\text{instance}, x) \in R\ \forall e \quad \text{distrib}_{r_P}(r, e, s) = \text{distrib}(\mathcal{S}(\text{instance},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

Problem with a Malicious Verifier

$$
\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
w \text{ st } R(x,w) & x & \\
\text{pick } r_P & & \\
a = \mathcal{P}(x,w;r_P) & \xrightarrow{\quad a \quad} & \text{take } e = f(x,a) \\
 & \xleftarrow{\quad e \quad} & \\
z = \mathcal{P}(x,w,e;r_P) & \xrightarrow{\quad z \quad} & V(x,a,e,z)?
\end{array}
$$

- 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



Prover **Verifier**

w st $R(x, w)$ x

pick r_P pick $e \in E$

$\xleftarrow{\text{Commit}(e;r)}$ pick r

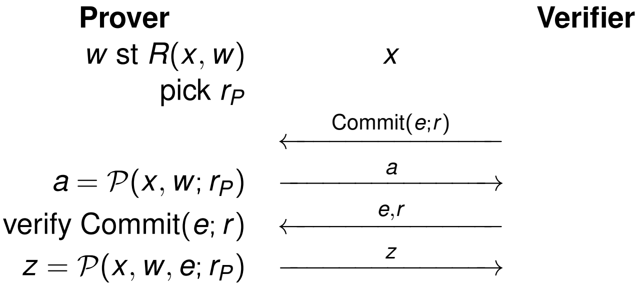
$a = \mathcal{P}(x, w; r_P)$ $\xrightarrow{a}$

verify Commit(e; r) $\xleftarrow{e,r}$

$z = \mathcal{P}(x, w, e; r_P)$ $\xrightarrow{z}$ $V(x, a, e, z)$?

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

Proving the ZK Property



Prover **Verifier**

w st $R(x,w)$ x

pick r_P

$\xleftarrow{\text{Commit}(e;r)}$

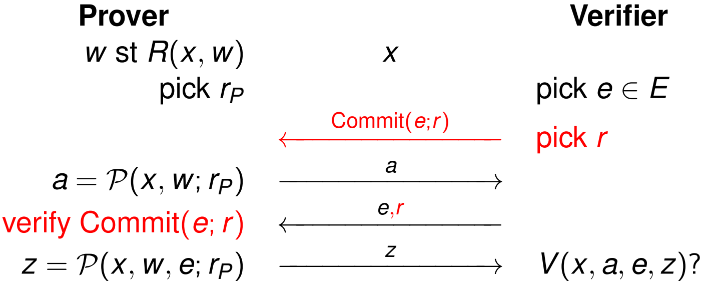
$a = \mathcal{P}(x,w;r_P)$ $\xrightarrow{a}$

verify Commit$(e;r)$ $\xleftarrow{e,r}$

$z = \mathcal{P}(x,w,e;r_P)$ $\xrightarrow{z}$

- 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



Prover

Verifier

w st $R(x, w)$

x

pick r_P

pick $e \in E$

$\xleftarrow{\text{Commit}(e;r)}$

pick r

$a = \mathcal{P}(x, w; r_P)$ $\xrightarrow{a}$

verify $\text{Commit}(e; r)$ $\xleftarrow{e,r}$

$z = \mathcal{P}(x, w, e; r_P)$ $\xrightarrow{z}$ $V(x, a, e, z)$?

- $*$-ZK (commitment $*$-binding)
- 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 $g \in \mathbf{Z}_p^*$ of order q, $\tau \in \mathbf{Z}_q^*$, and $h = g^{\tau} \bmod p$

  Domain parameters: $\langle p, q, g, h \rangle$
- **commit**: $\mathrm{Commit}(X; r) = g^X h^r \bmod p$
- **unconditionally hiding**
- **computationally binding**
- **trapdoor**

Pedersen Commitment — i

$$h = g^{\tau} \bmod p \qquad \mathsf{Commit}(X; r) = g^X h^r \bmod p$$

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

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

so c and X are statistically independent (perfect secrecy)

Pedersen Commitment — ii

$$h = g^{\tau} \bmod p \qquad \mathsf{Commit}(X; r) = g^X h^r \bmod p$$

- **computationally binding**: commiting 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

```
Setup$(1^s) \xrightarrow{\$} (p,q,g,h)$
$\mathcal{A}(p,q,g,h) \xrightarrow{\$} (x,r,x',r')$
return
    $1_{x \neq x', \mathsf{Commit}(x;r)=\mathsf{Commit}(x';r')}$
```

Game DL

```
Setup$(1^s) \xrightarrow{\$} (p,q,g)$
pick $\tau \in \mathbf{Z}_q$, $h \leftarrow g^{\tau}$
$\mathcal{B}(p,q,g,h) \xrightarrow{\$} z$
return $1_{h=g^z}$
```

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

```
$\mathcal{A}(p,q,g,h) \xrightarrow{\$} (x,r,x',r')$
$z \leftarrow \frac{x'-x}{r-r'} \bmod q$
return $z$
```

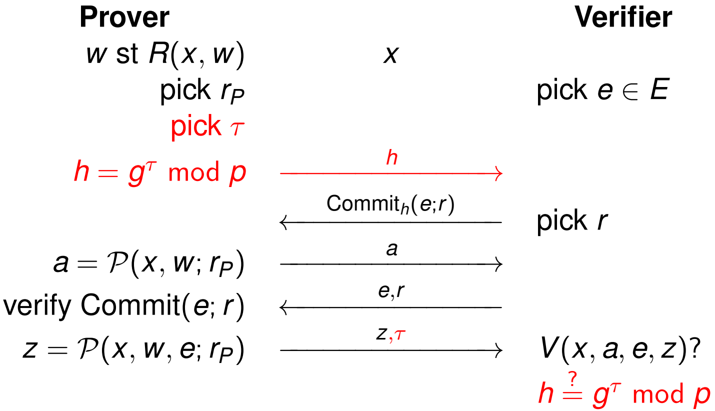
Pedersen Commitment — iii

$$h = g^{\tau} \bmod p \qquad \mathrm{Commit}(X; r) = g^X h^r \bmod p$$

- **trapdoor**: using τ we can open a commitment $\mathrm{Commit}(X_0; r_0)$ on an arbitrary X:

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

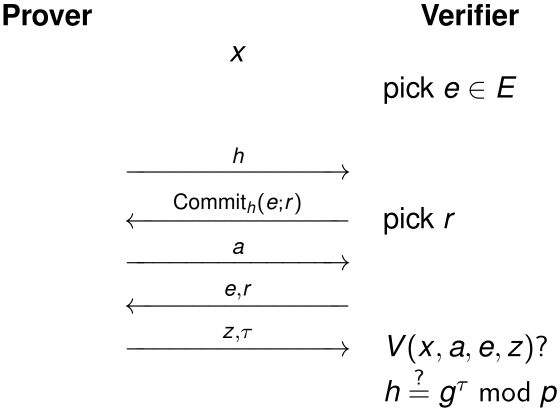
Strengthened Protocol



Prover		Verifier
w st $R(x,w)$	x	
pick r_P		pick $e \in E$
pick τ		
$h = g^\tau \bmod p$	$\xrightarrow{h}$	
	$\xleftarrow{\text{Commit}_h(e;r)}$	pick r
$a = \mathcal{P}(x,w;r_P)$	$\xrightarrow{a}$	
verify Commit$(e;r)$	$\xleftarrow{e,r}$	
$z = \mathcal{P}(x,w,e;r_P)$	$\xrightarrow{z,\tau}$	$V(x,a,e,z)$?
		$h \overset{?}{=} g^\tau \bmod p$

- now computational ZK (based on the hardness of discrete logarithm)
- sound

Proving Soundness



Prover **Verifier**

x

pick $e \in E$

$\xrightarrow{h}$

$\xleftarrow{\text{Commit}_h(e;r)}$ pick r

$\xrightarrow{a}$

$\xleftarrow{e,r}$

$\xrightarrow{z,\tau}$ $V(x,a,e,z)$?

$h \overset{?}{=} g^\tau \bmod p$

- run $\mathcal{P}^*$ once on r_P and simulate $\mathcal{V}$ with commitment c
- get τ at the end
- rewind $\mathcal{P}^*$ and run it again on r_P with same c
- open c on e' (using τ) as in the standard extraction

1. The Cryptographic Zoo

2. Cryptographic Security Models

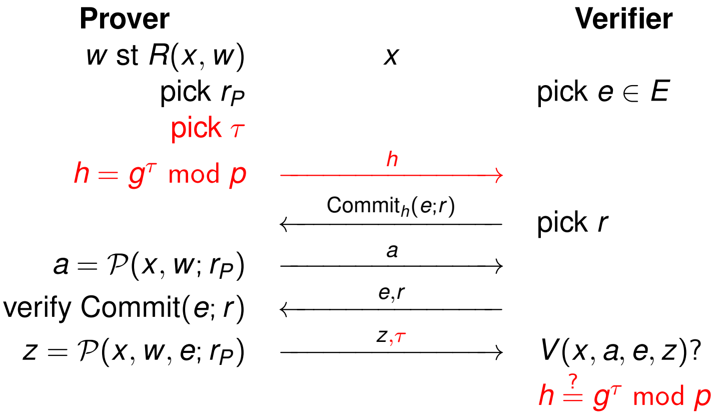
3. Cryptanalysis (Public-Key)

4. **The Power of Interaction**
   - Interactive Proofs
   - Zero-Knowledge
   - Zero-Knowledge Construction from Σ Protocol
   - **Setup Models**
   - A Building Block for Making Cryptographic Primitives

5. Cryptanalysis (Conventional)

6. Proving Security

ZK from Σ-Protocol Again



Prover		**Verifier**
w st $R(x, w)$	x	
pick r_P		pick $e \in E$
pick τ		
$h = g^\tau \bmod p$	$\xrightarrow{h}$	
	$\xleftarrow{\text{Commit}_h(e;r)}$	pick r
$a = \mathcal{P}(x, w; r_P)$	$\xrightarrow{a}$	
verify Commit$(e; r)$	$\xleftarrow{e,r}$	
$z = \mathcal{P}(x, w, e; r_P)$	$\xrightarrow{z,\tau}$	$V(x, a, e, z)$?
		$h \overset{?}{=} g^\tau \bmod p$

Why not setting up h once for all as a **common reference string** and give τ to the extractor?

Common Reference String Model

CRS in Theory

- **setup**: generate a pair (crs, τ) made of a common reference string crs and a trapdoor τ
- add crs as an input to $\mathcal{P}$, $\mathcal{V}$
- add τ 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 Σ-Protocol Again

Prover		**Verifier**
w st $R(x,w)$	x	
pick r_P		pick $e \in E$
pick τ		
$h = g^\tau \bmod p$	$\xrightarrow{h}$	pick r
	$\xleftarrow{c}$	$c = g^e h^r \bmod p$
$a = \mathcal{P}(x,w;r_P)$	$\xrightarrow{a}$	
$c \stackrel{?}{=} g^e h^r \bmod p$	$\xleftarrow{e,r}$	
$z = \mathcal{P}(x,w,e;r_P)$	$\xrightarrow{z,\tau}$	$V(x,a,e,z)$?
		$h \stackrel{?}{=} g^\tau \bmod p$

- 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

Prover		**Verifier**
w st $R(x, w)$	x	
pick r_P		pick $e \in E$
		pick r
	$\xleftarrow{\quad c \quad}$	$c = H(e \Vert r)$
$a = \mathcal{P}(x, w; r_P)$	$\xrightarrow{\quad a \quad}$	
$c \stackrel{?}{=} H(e \Vert r)$	$\xleftarrow{\quad e,r \quad}$	
$z = \mathcal{P}(x, w, e; r_P)$	$\xrightarrow{\quad z \quad}$	$V(x, a, e, z)$?

- 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 Σ Protocol
   - Setup Models
   - A Building Block for Making Cryptographic Primitives

5. Cryptanalysis (Conventional)

6. Proving Security

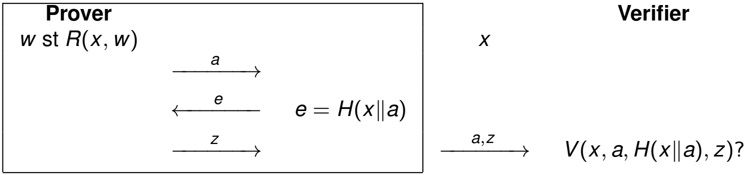
Constructions

- non-interactive zero-knowledge proofs (NIZK) from Σ-protocol
- signature from Σ-protocol
- trapdoor commitment from Σ-protocol
- hash function from Σ-protocol (in exercise)

The Fiat-Shamir Paradigm 1986

NIZK from Σ-Protocol

(making the Σ verifier malicious by selecting e adaptively...)



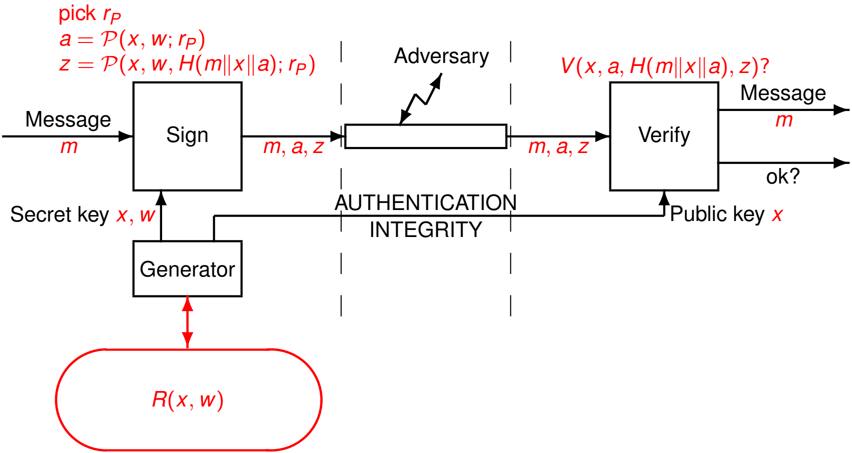
- replace $\mathcal{V}(x,a;r_V)$ by a random oracle $H(x\|a)$
- the final view is a non-interactive proof
- a random oracle “looks like” a honest verifier
  $\rightarrow$ “kind of” zero-knowledge
- a simulator for the final view becomes a cheating prover
  $\rightarrow$ cannot be zero-knowledge and sound at the same time

Fiat-Shamir Signature

Signature from Σ-Protocol

- public key: x
- secret key: (x, w)
- signature:
  1. pick r_P, set $a = \mathcal{P}(x, w; r_p)$
  2. replace $\mathcal{V}(x, a; r_V)$ by $e = H(\text{message}\|x\|a)$
  3. set $z = \mathcal{P}(x, w, e; r_P)$
  4. signature is the pair (a, z)
- verification: $V(x, a, H(\text{message}\|x\|a), z)$

Fiat-Shamir Signature



Full Fiat-Shamir Signature

Prover		**Verifier**
s st $s^2 v \bmod n = 1$	(n, v)	
pick $r_i \in \mathbf{Z}_n^*$		pick $e_i \in \{0, 1\}$
$x_i = r_i^2 \bmod n$	$\xrightarrow{x_1,\ldots,x_t}$	
	$\xleftarrow{e_1,\ldots,e_t}$	
$y_i = r_i s^{e_i} \bmod n$	$\xrightarrow{y_1,\ldots,y_t}$	$y_i^2 v^{e_i} \bmod n \overset{?}{=} x_i$

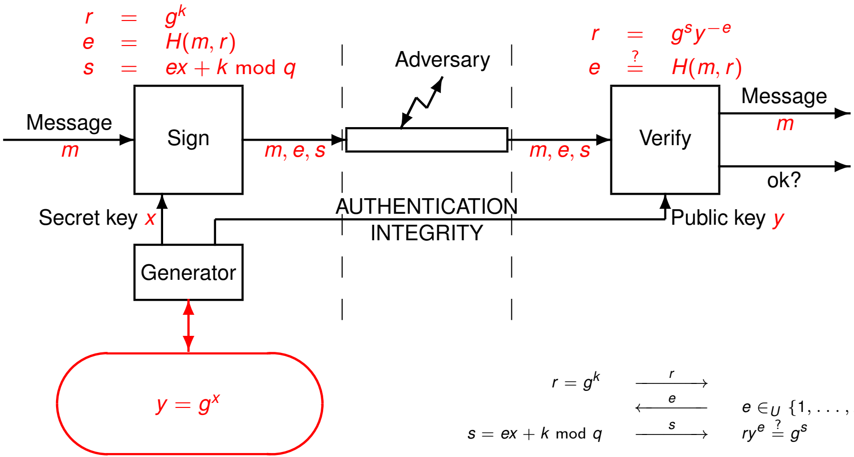
- domain parameter: n
- public key: v
- secret key: s such that $s^2 v \bmod n = 1$
- signature: pick $\vec{r}$, set $x_i = r_i^2 \bmod n$, $\vec{e} = H(\text{message} \| \vec{x})$,
  $y_i = r_i s^{e_i} \bmod n$
  signature is $(\vec{x}, \vec{y})$
- verification: $y_1^2 v^{e_1} \bmod n = x_1, \ldots, y_t^2 v^{e_t} \bmod n = x_t$ with
  $\vec{e} = H(\text{message} \| \vec{x})$

Schnorr Signature

$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
x \text{ st } g^x = y & (G, q, g, y) & \\
\text{pick } k \in \mathbf{Z}_q & & \text{pick } e \in \{1, \ldots, 2^t\} \\
r = g^k & \xrightarrow{\qquad r \qquad} & \\
& \xleftarrow{\qquad e \qquad} & \\
s = ex + k \bmod q & \xrightarrow{\qquad s \qquad} & ry^e \stackrel{?}{=} g^s
\end{array}$$

- 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



$$\begin{array}{ccc} r = g^k & \xrightarrow{r} & \\ & \xleftarrow{e} & e \in_U \{1, \ldots, 2^t\} \\ s = ex + k \bmod q & \xrightarrow{s} & ry^e \overset{?}{=} g^s \end{array}$$

The Security of a Fiat-Shamir Signature in ROM

Theorem

*Given a relation R s.t. it is hard to find witnesses and a Σ-protocol for its language s.t. $1/\#E = \mathrm{negl}$, 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 another chapter)

Trapdoor Commitment

Commitment on $e \in E$ based on a Σ-protocol such that finding a witness is hard

- **Setup**$\rightarrow (x, w)$: take a parameter (crs) $x \in L$ such that nobody knows a witness (trapdoor) w s.t. $(x, w) \in R$
- **Commit**$(x, e) \rightarrow (a, z)$: to commit to $e \in E$, pick r and let $(a, e, z) = \mathcal{S}(x, e; r)$
commit value is a
opening key is (e, z)
- **Open**$(x, a, e, z) \rightarrow 0/1$: check $V(x, a, e, z)$ holds
- **perfectly hiding** (distribution of a independent from e)
- **computationally binding** (otherwise extract w)
- **trapdoor** w: make (using $\mathcal{P}$) a commit value a which can open to 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.

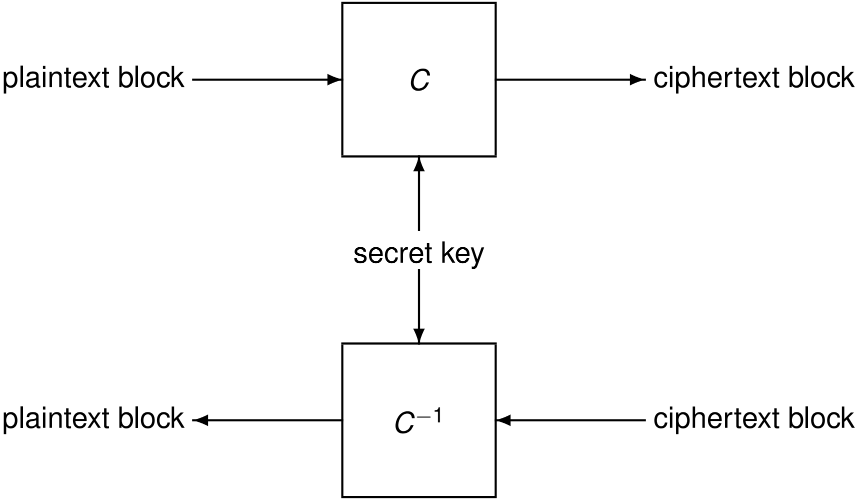
Train Yourself

- Σ-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)
  final exam 2020–21 ex3 (MPC-in-the-head)
  final exam 2021–22 ex1 (DLEQ)
  midterm exam 2022–23 ex2 (DLEQ)
- 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 Σ)
- weak Fiat-Shamir: final exam 2014–15 ex2
- Unruh transform: final exam 2017–18 ex3
- soundness of DLEQ: final exam 2023–24 ex1

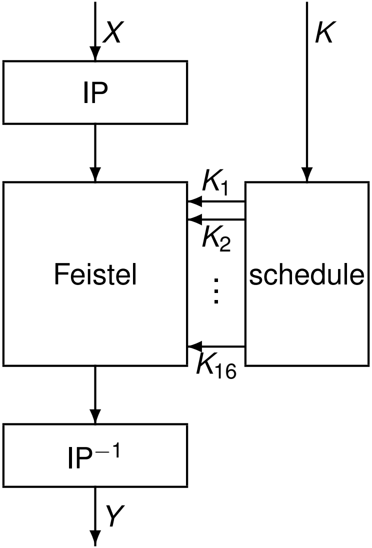
1. The Cryptographic Zoo
2. Cryptographic Security Models
3. Cryptanalysis (Public-Key)
4. The Power of Interaction
5. **Cryptanalysis (Conventional)**
6. Proving Security

1. The Cryptographic Zoo
2. Cryptographic Security Models
3. Cryptanalysis (Public-Key)
4. The Power of Interaction
5. **Cryptanalysis (Conventional)**
    - **Block Ciphers**
    - Differential Cryptanalysis
    - Linear Cryptanalysis
    - Hypothesis Testing in Cryptography
    - Decorrelation
6. Proving Security

Block Cipher

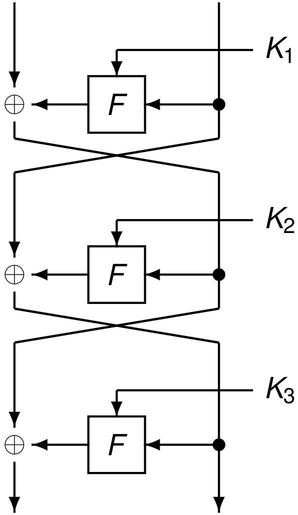


Example: DES



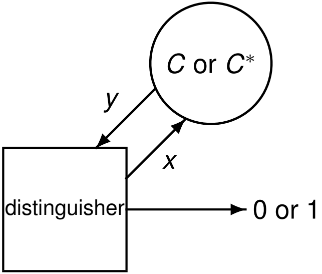
DES — Feistel Scheme

$\Psi(F^{K_1}, F^{K_2}, F^{K_3})$



Distinguishing Attack on Ciphers

Indinstinguishability 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 Kth permutation of $\{0,1\}^\ell$.

Reminder (Stirling Formula)

$$n! \sim \sqrt{2\pi n} n^n e^{-n}$$

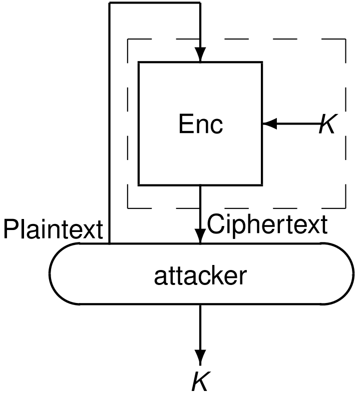
Number of bits to represent K: $\log_2(2^\ell!) \approx \ell 2^\ell$...
Example for $\ell = 64$: $\log_2(2^\ell!) \approx 1\,180\,591\,620\,717\,411\,303\,424$ bits
(1 048 576 Petabytes)

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

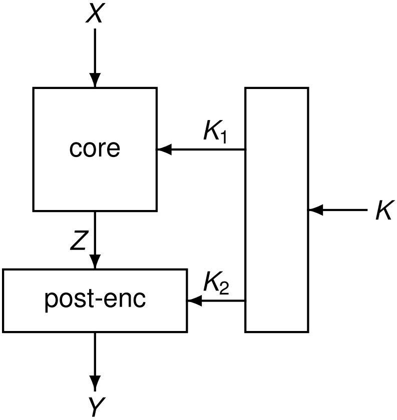
- invented by Eli Biham and Adi Shamir (Biham's PhD Thesis)
- 1990: broke DES-like ciphers
- 1992: theoretical attack against DES requiring 2^{47} chosen plaintexts (not realistic enough to be called an attack)
- 1993: breakdown if the design of DES is slightly modified
- 1994: Coppersmith claimed that 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

$$\Pr[\Delta Z = b | \Delta X = a] \text{ large}$$

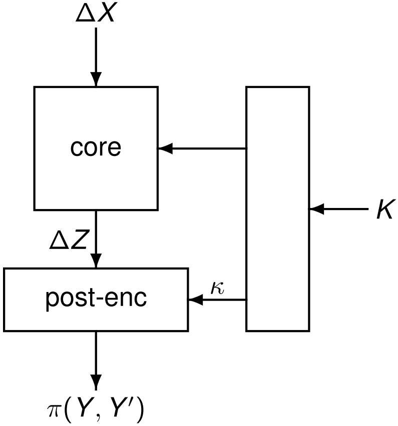
Difference: $\Delta Z = Z' \oplus Z$

- 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 $\Delta Z = b$):
"ΔZ is consistent with κ and $\pi(Y, Y')$"



Step 4: Implementation

Precomputation:

```
initialize SubCandidate$_u$ to empty set for all $u$
for all $u$ and all $\kappa$ such that $R(\kappa, u)$, insert $\kappa$ in SubCandidate$_u$
```

Collection phase:

```
collect $n$ pairs $((x, y), (x \oplus a, y'))$ of plaintext-ciphertext pairs
```

Analysis phase:

```
initialize counters $m_\kappa$ to 0
for each pair do
    compute $u = \pi(y, y')$
    for all $\kappa \in$ SubCandidate$_u$ increment $m_\kappa$
end for
sort all possible $\kappa$ in decreasing order of $m_\kappa$
```

Search phase:

```
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

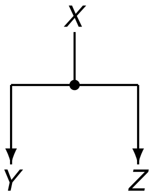
$$\mathrm{DP}^f(a,b) = \Pr_X[f(X \oplus a) = f(X) \oplus b]$$

where $X \in_U \{0,1\}^p$.

- Property: $\mathrm{DP}^f(0,b) = \begin{cases} 1 & \text{if } b = 0 \\ 0 & \text{otherwise} \end{cases}$
- Property: for all a, $\displaystyle\sum_{b \in \{0,1\}^q} \mathrm{DP}^f(a,b) = 1$
- Property: $2^p.\mathrm{DP}^f(a,b)$ is even

Differential Circuit — i: Duplicate Gate

computation circuit

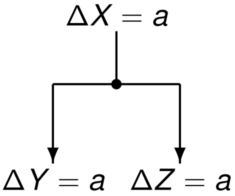


X

Y Z

$$X = Y = Z$$

differential circuit



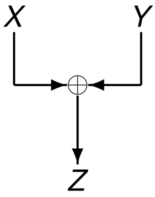
$\Delta X = a$

$\Delta Y = a$ $\Delta Z = a$

$$\Delta X = a \Longrightarrow \Delta Y = \Delta Z = a$$

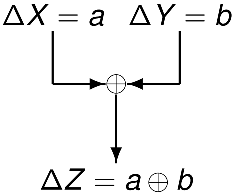
Differential Circuit — ii: XOR Gate

computation circuit



$$Z = X \oplus Y$$

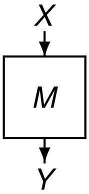
differential circuit



$$(\Delta X = a, \Delta Y = b) \Longrightarrow \Delta Z = a \oplus b$$

Differential Circuit — iii: Linear Circuit

computation circuit



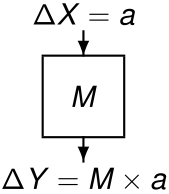
X

M

Y

$$Y = M \times X$$

differential circuit



$\Delta X = a$

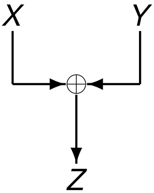
M

$\Delta Y = M \times a$

$$\Delta X = a \Longrightarrow \Delta Y = M \times a$$

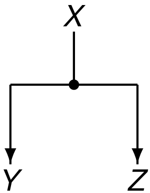
Addition and Duplicate Gates

XOR Gate



$$\begin{pmatrix} Z \end{pmatrix} = \begin{pmatrix} 1 & 1 \end{pmatrix} \times \begin{pmatrix} X \\ Y \end{pmatrix}$$

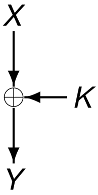
Duplicate Gate



$$\begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \times \begin{pmatrix} X \end{pmatrix}$$

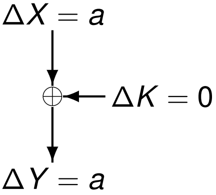
Differential Circuit — iv: XOR to Constant Gate

computation circuit



$Y = X \oplus K$

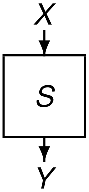
differential circuit



$\Delta X = a \Longrightarrow \Delta Y = a$

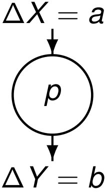
Differential Circuit — v: Non-Linear Circuit

computation circuit



$$Y = S(X)$$

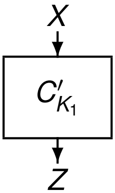
differential circuit



$$\Pr[\Delta Y = b | \Delta X = a] = p$$

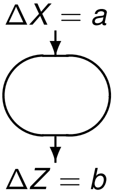
Differential Characteristic

computation circuit



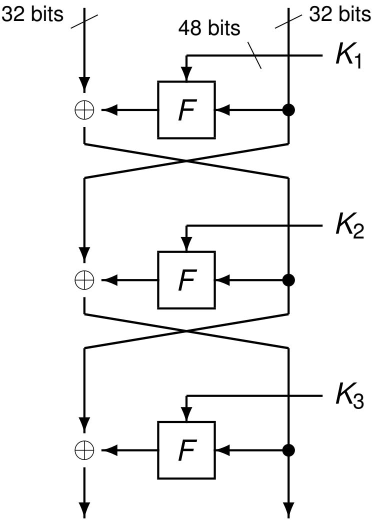
$$Z = C'_{K_1}(X)$$

differential circuit

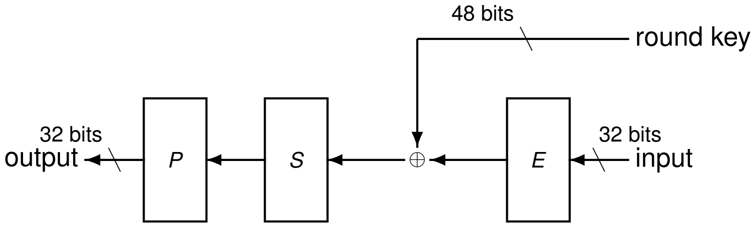


$$\Pr[\Delta Z = b | \Delta X = a] = \mathsf{DP}^{C'_{K_1}}(a, b)$$

DES (Reminder) — i

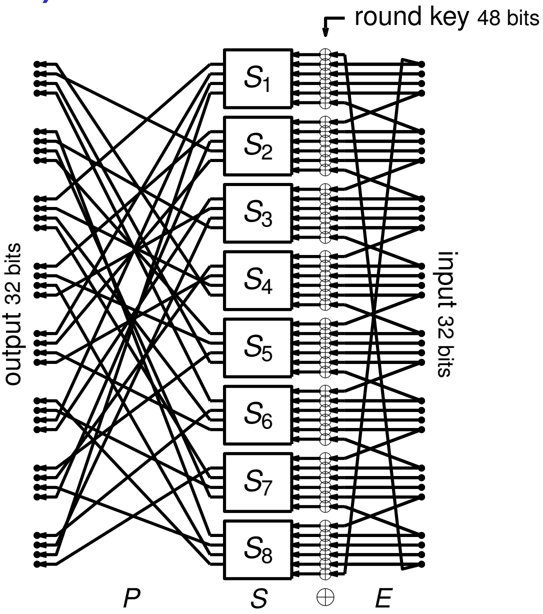


DES (Reminder) — ii

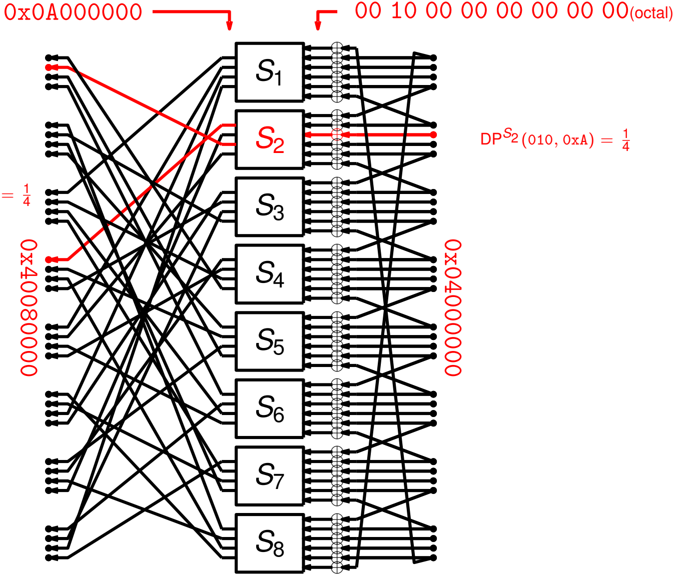


- E: expansion (duplicate one bit out of two)
- $\oplus$: bitwise XOR with a round key
- S: substitution boxes (eight 6-to-4 bits mappings)
- P: permutation of bits

DES (Reminder) — iii



Differential Cryptanalysis of 8R-DES



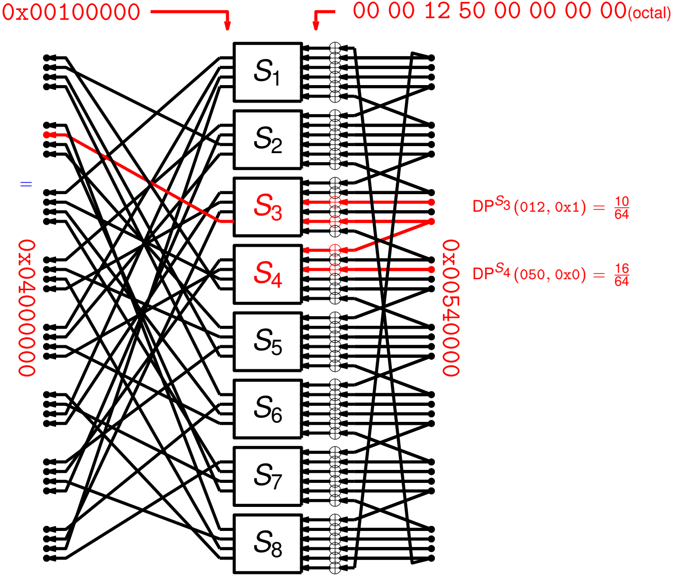
$\mathrm{DP}^{S_2}(\mathtt{010}, \mathtt{0xA}) = \frac{1}{4}$

red sub-characteristic:
$\mathrm{DP}^{F}(\mathtt{0x04000000}, \mathtt{0x40080000}) = \frac{1}{4}$

$$p = 1 \cdot \frac{1}{4} \cdot 1 \cdots 1$$

Other Sub-Characteristic

blue sub-characteristic:
$\mathrm{DP}^F(\mathtt{0x00540000}, \mathtt{0x04000000}) = \frac{5}{128}$

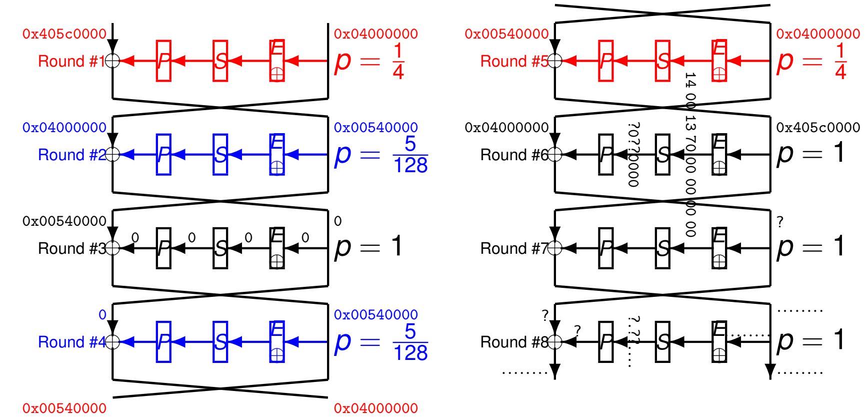


$\mathrm{DP}^{S_3}(\mathtt{012}, \mathtt{0x1}) = \frac{10}{64}$

$\mathrm{DP}^{S_4}(\mathtt{050}, \mathtt{0x0}) = \frac{16}{64}$

$$p = 1 \cdot 1 \cdot \frac{10}{64} \cdot \frac{16}{64} \cdot 1 \cdots 1$$

Differential Characteristic



$$a = \texttt{0x405c0000 0x04000000} \qquad b = \texttt{0x04000000 0x405c0000}$$

$$\mathrm{DP}^{\mathrm{core}}(a, b) \approx 2^{-13.4}$$

Ciphertext Pair Analysis (Predicate R)

Theorem

It the characteristic is satisfied (i.e., $\Delta Z = b$), then

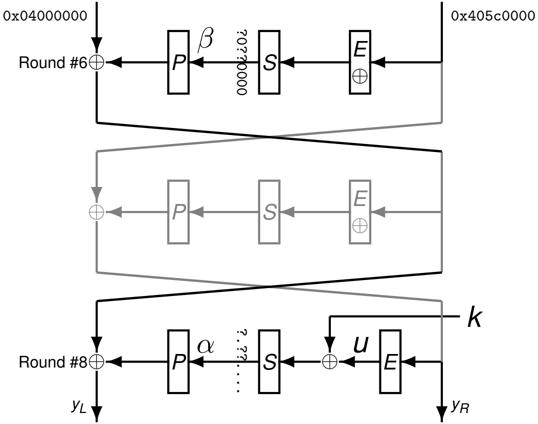
$$\left(P^{-1}(\Delta y_L \oplus 04000000)\right)_i = S_i(k_i \oplus E(y_R)_i) \oplus S_i(k_i \oplus E(y'_R)_i)$$

for $i = 2, 5, 6, 7, 8$

so, we can let

$$\kappa = (k_2, k_5, k_6, k_7, k_8)$$

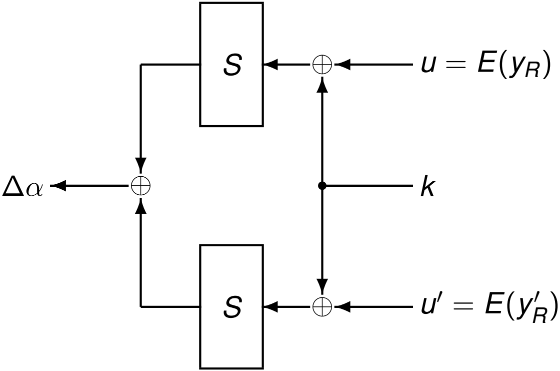
Proof — i



- $\Delta y_L = 04000000 \oplus P(\Delta\beta \oplus \Delta\alpha)$, $\Delta\beta_i = 0$ for $i = 2, 5, 6, 7, 8$
- so, $\left(P^{-1}(\Delta y_L \oplus 04000000)\right)_i = \Delta\alpha_i$ for $i = 2, 5, 6, 7, 8$

Proof — ii

Let $u = E(y_R)$, $u' = E(y'_R)$, $v = P^{-1}(\Delta y_L \oplus \texttt{04000000})$,
$\Delta\alpha = S(k \oplus u) \oplus S(k \oplus u')$



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

$$v = P^{-1}(y_L \oplus y'_L \oplus \texttt{04000000})$$

  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

$$\text{SubCandidate}_{i,\mu,\mu',\nu} = \{r \in \{0,1\}^6; \nu = S_i(\mu \oplus r) \oplus S_i(\mu' \oplus r)\}$$

- 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 $\mathrm{SubCandidate}_{i,\mu,\mu',\nu}$ 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
   $\mathrm{SubCandidate}_{i,\mu,\mu',S_i(\mu \oplus r) \oplus S_i(\mu' \oplus r)}$
```

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 $k_i \in \mathrm{SubCandidate}_{i,u_i,u'_i,v_i}$ 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

- $E(X_1 + \cdots + X_n) = E(X_1) + \cdots + E(X_n)$
- $V(X_1 + \cdots + X_n) = V(X_1) + \cdots + V(X_n)$ when $X_1, \ldots, X_n$ are independent
- $V(X) = E(X)(1 - E(X))$ when X is Boolean (support in $\{0, 1\}$)
- informally, for $X_1, \ldots, X_n$ iid Boolean of exp. value μ we have
$$X_1 + \cdots + X_n \approx n\mu \pm \sqrt{n\mu(1 - \mu)}$$

Complexity Analysis (Heuristic)

- The right 30-bit subkey candidate is suggested once per iteration with probability $p_1 = \mathrm{DP} = 2^{-13.4}$

$$\mathrm{Signal} = np_1 \pm \sqrt{np_1}$$

- Each iteration suggests 4^5 key candidates randomly, so every candidate is suggested once per iteration with probability $p_2 = 2^{-20}$

$$\mathrm{Noise} = np_2 \pm \sqrt{np_2}$$

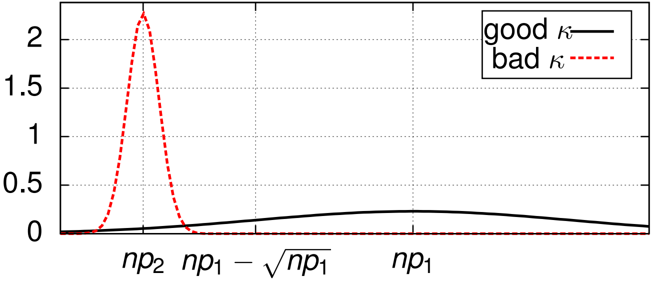
- We need n such that $\sqrt{np_1} \ll n(p_1 - p_2) \approx np_1$

$$n \gg \frac{1}{\mathrm{DP}}$$

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

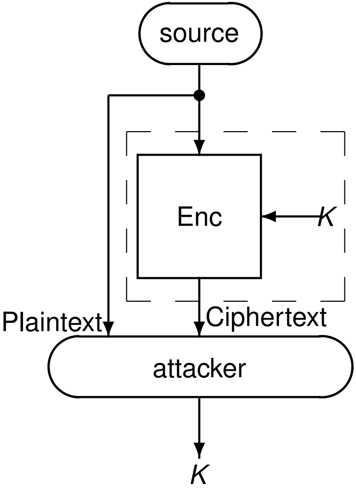
5. **Cryptanalysis (Conventional)**
   - Block Ciphers
   - Differential Cryptanalysis
   - Linear Cryptanalysis
   - Hypothesis Testing in Cryptography
   - Decorrelation

6. Proving Security

History

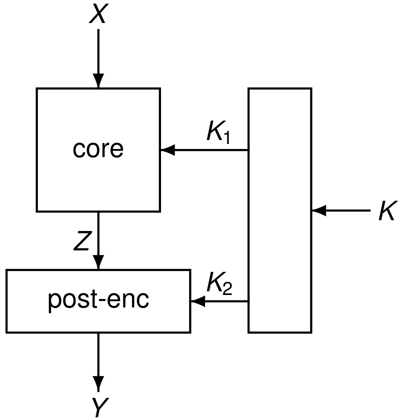
- 1977 DES: an unpopular standard with secret rationales
- 1987 FEAL: a Japanese version
- 1990 Biham-Shamir: differential cryptanalysis
- 1990 Gilbert et al.
- 1993 Matsui: linear cryptanalysis
- 1994 Matsui: application to DES (requires 2^{43} known plaintexts)

From Chosen to Known Plaintext Key Recovery



Step 1: Cipher Decomposition

find an appropriate decomposition of following form



(could use pre-encryption as well)

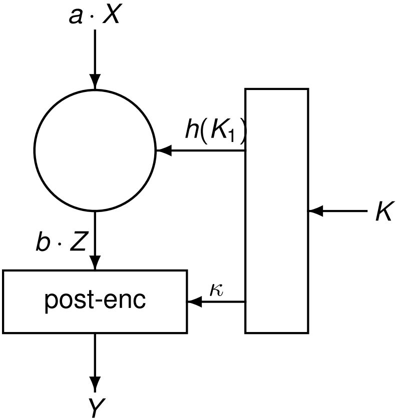
Step 2: Deviant Property

$$\left|\Pr\left[X_{i_1} \oplus \cdots \oplus X_{i_r} = Z_{j_1} \oplus \cdots \oplus Z_{j_s}\right] - \frac{1}{2}\right| \text{ large}$$

$$\begin{array}{lll} X_{i_1} \oplus \cdots \oplus X_{i_r} & \text{can be written} & a_1X_1 \oplus \cdots \oplus a_pX_p = a \cdot X \\ Z_{j_1} \oplus \cdots \oplus Z_{j_s} & \text{can be written} & b_1Z_1 \oplus \cdots \oplus b_qZ_q = b \cdot Z \end{array}$$

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:

```
for all possible $u = \pi(X, Y)$ do
    initialize a counter $n_u$ to zero
end for
collect $n$ Plaintext-Ciphertext pairs $(X, Y)$
for each $(X, Y)$ do
    compute $u = \pi(X, Y)$
    increment $n_u$
end for
```

Analysis phase:

```
for all possible $\kappa$ do
    compute
        $$m_\kappa = \sum_{u \text{ s.t. } P(\kappa,u)=0} n_u$$
end for
sort all $\kappa$ in decreasing order of $|m_\kappa - \frac{n}{2}|$
```

Search phase:

```
for each sorted $\kappa$ exhaustively look for $K$
note: $h(K_2)$ is likely to be $1_{m_\kappa < n/2}$
```

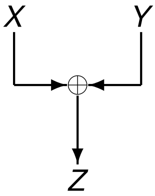
Making Linear Characteristics [Biham,Eurocrypt 94]

Goal: compute $b \cdot Z = (a \cdot X) \oplus$ 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.

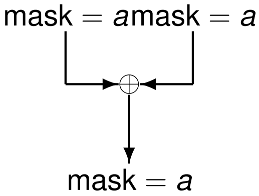
Dual Circuit — i: XOR Gate

computation circuit



$$Z = X \oplus Y$$

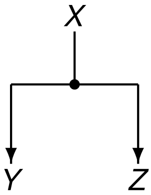
mask circuit



$$a \cdot Z = (a \cdot X) \oplus (a \cdot Y)$$

Dual Circuit — ii: Duplicate Gate

computation circuit

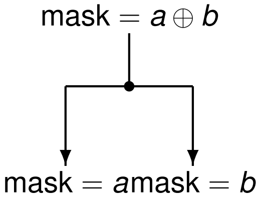


X

Y Z

$$X = Y = Z$$

mask circuit



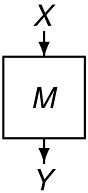
mask $= a \oplus b$

mask $= a$mask $= b$

$$(a \cdot Y) \oplus (b \cdot Z) = (a \oplus b) \cdot X$$

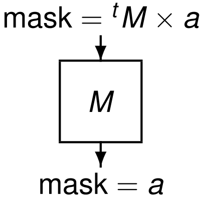
Dual Circuit — iii: Linear Circuit

computation circuit



$$Y = M \times X$$

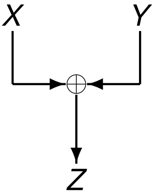
mask circuit



$$a \cdot Y = ({}^tM \times a) \cdot X$$

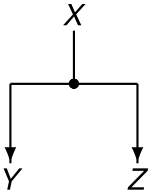
Addition and Duplicate Gates

XOR Gate



$$\left(Z \right) = \left(1 \quad 1 \right) \times \begin{pmatrix} X \\ Y \end{pmatrix}$$

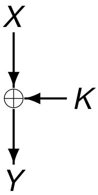
Duplicate Gate



$$\begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \times \left(X \right)$$

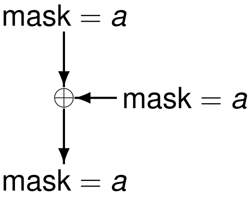
Dual Circuit — iv: XOR to Constant Gate

computation circuit



$$Y = X \oplus K$$

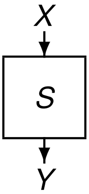
mask circuit



$$a \cdot Y = (a \cdot X) \oplus \underbrace{(a \cdot K)}_{\text{constant}}$$

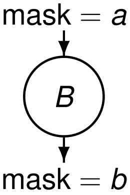
Dual Circuit — v: Non-Linear Circuit

computation circuit



$$Y = S(X)$$

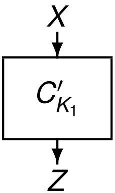
mask circuit



$$b \cdot Y = a \cdot X \oplus \underbrace{B}_{\text{biased}}$$

Linear Characteristic

computation circuit



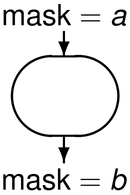
X

C'_{K_1}

Z

$$Z = C'_{K_1}(X)$$

mask circuit



$\text{mask} = a$

$\text{mask} = b$

$$(a \cdot X) \oplus (b \cdot Z) = \text{bit}(K_1) \oplus \bigoplus_{i=1}^{n} B_i$$

Piling-up Lemma

Lemma (Piling up Lemma)

For any Boolean variable B we define $\mathrm{LP}(B) = (2\Pr[B = 0] - 1)^2$. *Let $B_1, \ldots, B_n$ be n independent random variables. We have*

$$\mathrm{LP}(B_1 \oplus \ldots \oplus B_n) = \mathrm{LP}(B_1) \times \ldots \times \mathrm{LP}(B_n).$$

Proof. Note that

- $\mathrm{LP}(B) = \left(E\left((-1)^B\right)\right)^2$,
- $(-1)^{A \oplus B} = (-1)^A \times (-1)^B$,
- $E(X \times Y) = E(X) \times E(Y)$ 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

$$\mathrm{LP}^f(a,b) = \left(2.\Pr_X[a \cdot X = b \cdot f(X)] - 1\right)^2$$

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: $\mathrm{LP}^f(a,b) = \mathrm{LP}((a \cdot X) \oplus (b \cdot f(X)))$

Link Between DPs and LPs

Theorem

Given a function f from $\{0,1\}^p$ to $\{0,1\}^q$ we have

$$\begin{aligned}
\mathrm{LP}^f(\alpha,\beta) &= \left(2\Pr[\alpha\cdot X=\beta\cdot f(X)]-1\right)^2\\
\mathrm{DP}^f(a,b) &= \Pr[f(X\oplus a)\oplus f(X)=b]\\
&= 2^{-q}\sum_{\alpha,\beta}(-1)^{a\cdot\alpha\oplus b\cdot\beta}\mathrm{LP}^f(\alpha,\beta)
\end{aligned}$$

Proof. First observe that $\mathrm{LP}^f(\alpha,\beta)=\left(E\left((-1)^{(\alpha\cdot X)\oplus(\beta\cdot f(X))}\right)\right)^2=E\left((-1)^{(\alpha\cdot(X\oplus Y))\oplus(\beta\cdot(f(X)\oplus f(Y)))}\right)$ then

$$\begin{aligned}
\sum_{\alpha,\beta}(-1)^{a\cdot\alpha\oplus b\cdot\beta}\mathrm{LP}^f(\alpha,\beta) &= E\left(\sum_{\alpha,\beta}(-1)^{(\alpha\cdot(a\oplus X\oplus Y))\oplus(\beta\cdot(b\oplus f(X)\oplus f(Y)))}\right)\\
&= 2^{p+q}E\left(1_{X\oplus Y=a,f(X)\oplus f(Y)=b}\right)\\
&= 2^q\mathrm{DP}^f(a,b)
\end{aligned}$$

□

Link Between DPs and LPs

Theorem

Given a function f from $\{0,1\}^p$ to $\{0,1\}^q$ we have

$$\begin{aligned}
\mathrm{DP}^f(a,b) &= \Pr\left[f(X\oplus a)\oplus f(X)=b\right]\\
\mathrm{LP}^f(\alpha,\beta) &= \left(2\Pr\left[\alpha\cdot X=\beta\cdot f(X)\right]-1\right)^2\\
&= 2^{-p}\sum_{a,b}(-1)^{a\cdot\alpha\oplus b\cdot\beta}\mathrm{DP}^f(a,b)
\end{aligned}$$

Proof.

$$\begin{aligned}
\sum_{a,b}(-1)^{a\cdot\alpha\oplus b\cdot\beta}\mathrm{DP}^f(a,b) &= 2^{-q}\sum_{a,b}\sum_{\alpha',\beta'}(-1)^{a\cdot\alpha\oplus b\cdot\beta}(-1)^{a\cdot\alpha'\oplus b\cdot\beta'}\mathrm{LP}^f(\alpha',\beta'\\
&= 2^{-q}\sum_{\alpha',\beta'}\sum_{a,b}(-1)^{a\cdot(\alpha\oplus\alpha')\oplus b\cdot(\beta\oplus\beta')}\mathrm{LP}^f(\alpha',\beta')\\
&= 2^p\sum_{\alpha',\beta'}1_{\alpha'=\alpha,\beta'=\beta}\mathrm{LP}^f(\alpha',\beta')\\
&= 2^p\mathrm{LP}^f(\alpha,\beta)
\end{aligned}$$

□

Application to DES8

$$\begin{tikzcd}
\mathtt{0x00008000} & \mathtt{0x40000000} \arrow[l, "P"'] \\
{} & \mathtt{20\ 00\ 00\ 00\ 00\ 00\ 00\ 00} \arrow[l, "S\ \left(\mathrm{LP}^{S_1}(\mathtt{020},\mathtt{4})=\left(\frac{2}{32}\right)^2\right)"'] \\
{} & \mathtt{0x20000000} \arrow[l, "E"']
\end{tikzcd}$$

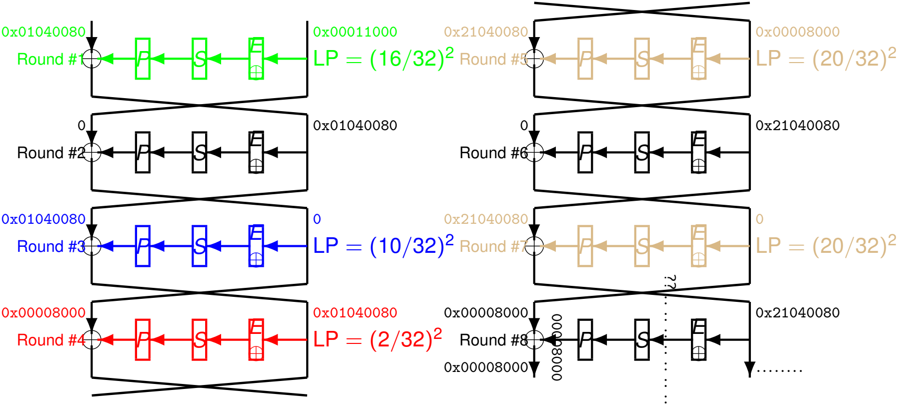
$$\begin{tikzcd}
\mathtt{0x01040080} & \mathtt{0x0000e000} \arrow[l, "P"'] \\
{} & \mathtt{00\ 00\ 00\ 00\ 42\ 00\ 00\ 00} \arrow[l, "S\ \left(\mathrm{LP}^{S_5}(\mathtt{042},\mathtt{e})=\left(\frac{16}{32}\right)^2\right)"'] \\
{} & \mathtt{0x00011000} \arrow[l, "E"']
\end{tikzcd}$$

Other Sub-Characteristics

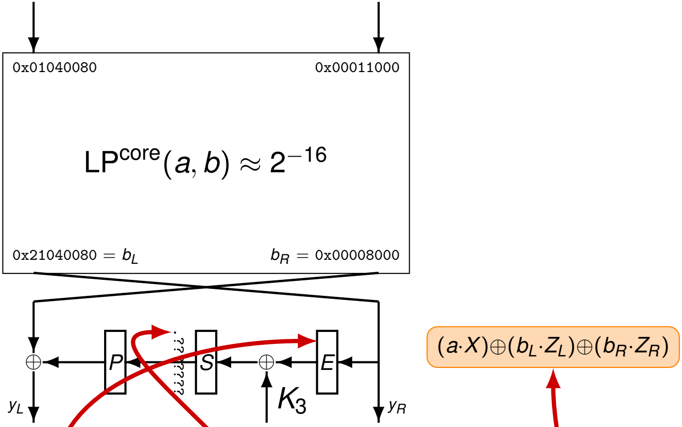
$$\begin{tikzcd}[column sep=huge]
\texttt{0x01040080} & \texttt{0x0000e000} \arrow[l, "P"'] \\
 & \texttt{00 00 00 00 20 00 00 00} \arrow[l, "S\ \left(\mathrm{LP}^{S_5}(\texttt{020},\texttt{e})=\left(\frac{10}{32}\right)^2\right)"'] \\
 & \texttt{0x00008000} \arrow[l, "E"']
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge]
\texttt{0x21040080} & \texttt{0x0000f000} \arrow[l, "P"'] \\
 & \texttt{00 00 00 00 20 00 00 00} \arrow[l, "S\ \left(\mathrm{LP}^{S_5}(\texttt{020},\texttt{f})=\left(\frac{20}{32}\right)^2\right)"'] \\
 & \texttt{0x00008000} \arrow[l, "E"']
\end{tikzcd}$$

Linear Characteristic



Projection



$(a{\cdot}X)\oplus(b_L{\cdot}Z_L)\oplus(b_R{\cdot}Z_R)$

- $a =$ `0x01040080 00011000`, $b =$ `0x21040080 00008000`, $\kappa = (K_3)_1$
- $\pi(x,y) = \begin{pmatrix} u = (E(y_R))_1 \\ v = (a \cdot x) \oplus (b_L \cdot y_R) \oplus (b_R \cdot y_L) \end{pmatrix}$
- $\mathrm{Project}(\kappa, (u,v)) = v \oplus (b_R \cdot P(S_1(\kappa \oplus u) \| \texttt{0x0000000}))$

Attack

Collection phase:

```
initialize $2^7$ counters $n_{u,v}$ to zero for all possible 6-bit values $u$ and all possible bits $v$.
collect $n$ plaintext-ciphertext $(x, y)$ pairs,
for each $(x, y)$ pair do
    set $u$ to the 6 leading bits of the expansion $E(y_R)$ of $y_R$ in the round function
    $v \leftarrow (a \cdot x) \oplus (b_L \cdot y_R) \oplus (b_R \cdot y_L)$
    increment $n_{u,v}$
end for
```

Analysis phase:

```
for all possible $\kappa$ do
    $m_\kappa \leftarrow \sum_u n_{u, b_R \cdot (P(S_1(u \oplus \kappa) \| 0 \ldots 0))}$
end for
sort all $\kappa$ in decreasing order of $|m_\kappa - \frac{n}{2}|$,
```

Search phase:

```
do an exhaustive search by using the sorted list for $\kappa$.
```

Analysis

Theorem

For $n \approx \frac{1}{\mathsf{LP}}$, the correct value κ is first in the sorted list with high probability.

Example: for 8-round DES, $\mathsf{LP} = 2^{-16}$

1 The Cryptographic Zoo

2 Cryptographic Security Models

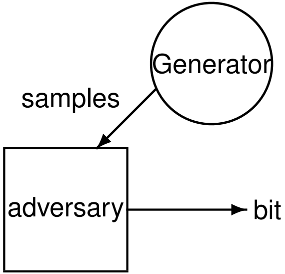
3 Cryptanalysis (Public-Key)

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, ε)-indistinguishable if for any algorithm $\mathcal{A}$ taking q iid random variables $x_1, \ldots, x_q$ following P we have

$$|\mathsf{Adv}_{\mathcal{A}}(P_0, P_1)| \leq \varepsilon$$

where

$$\mathsf{Adv}_{\mathcal{A}}(P_0, P_1) = \Pr[\mathcal{A} \to 1 | P = P_1] - \Pr[\mathcal{A} \to 1 | P = P_0]$$

A notion of distance between P_0 and P_1:

$$\mathsf{distance}_q(P_0, P_1) = \max_{\substack{\text{distinguisher}\\ \text{limited to q}}} |\Pr[\mathcal{A} \to 1 | P = P_1] - \Pr[\mathcal{A} \to 1 | P = P_0]|$$

Applications

- **pseudorandom number generator**
  break a PRNG $\Longrightarrow$ 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 $\{P_1, \ldots, P_n\}$

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: $\pi_0 = \pi_1 = \frac{1}{2}$

$$\begin{aligned} \mathrm{Adv}_{\mathcal{A}} &= \Pr[\mathcal{A} \rightarrow 1 | P_1] - \Pr[\mathcal{A} \rightarrow 1 | P_0] \\ \mathrm{Adv}_{\mathcal{A}} &= (1 - \beta) - \alpha \ = \ 1 - 2P_e \\ 1 - \mathrm{Adv}_{\mathcal{A}} &= \alpha + \beta \ = \ 2P_e \end{aligned}$$

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)
  $\rightarrow$ let $\mathcal{A}^{-1}(1)$ be the set of values x such that $\mathcal{A} \rightarrow 1$ when $X = x$
- we have
$$\mathrm{Adv}_{\mathcal{A}} = \sum_{x \in \mathcal{A}^{-1}(1)} (P_1(x) - P_0(x))$$
- clearly
$$\mathrm{Adv}_{\mathcal{A}} \leq \sum_{x; P_0(x) \leq P_1(x)} (P_1(x) - P_0(x))$$
- 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*

$$\mathrm{Adv}_{\mathcal{A}} \leq d(P_0, P_1)$$

Best Distinguisher (Single Sample)

```
input: $x$
$R = \frac{P_0(x)}{P_1(x)}$
if $R \leq 1$ then
    $b \leftarrow 1$
else
    $b \leftarrow 0$
end if
output: $b$
```

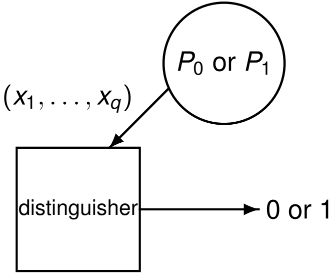
- R is the *likelihood ratio*

$$\mathrm{Adv}_{\mathcal{A}} = d(P_0, P_1)$$

- caveat: $\frac{p}{0} = +\infty$
- remark: $\frac{0}{0}$ never occurs

General Case

trick: consider $X = (X_1, \ldots, X_q)$ 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:

$$\text{output } 1 \Longleftrightarrow \frac{\Pr_{P_0^{\otimes q}}[x_1, \ldots, x_q]}{\Pr_{P_1^{\otimes q}}[x_1, \ldots, x_q]} \leq 1$$

Best Distinguisher (Multiple Samples)

```
input: $x_1, \ldots, x_q$
$R = \frac{P_0(x_1)\times\cdots\times P_0(x_q)}{P_1(x_1)\times\cdots\times P_1(x_q)} = \prod_x \frac{P_0(x)^{q_x}}{P_1(x)^{q_x}}$
        with $q_x = \#\{i; x_i = x\}$
if $R \leq 1$ then
    $b \leftarrow 1$
else
    $b \leftarrow 0$
end if
output: $b$
```

Example i: Biased Coin

$$P_0 = \text{uniform} \qquad P_1 = \begin{pmatrix} \text{head} & \text{tail} \\ 1 & 2 \\ \downarrow & \downarrow \\ \frac{1}{2}(1+\varepsilon) & \frac{1}{2}(1-\varepsilon) \end{pmatrix}$$

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

$$\text{output } 1 \iff \frac{1}{(1+\varepsilon)^{q_1}(1-\varepsilon)^{q_2}} \leq 1$$

$$\iff q_1 \log(1+\varepsilon) + q_2 \log(1-\varepsilon) \geq 0 \quad \overset{\sim}{\iff} \quad q_2 < q_1$$

Example ii: Biased Dice

$$P_0 = \text{uniform} \qquad P_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \frac{1}{6}+\varepsilon & \frac{1}{6} & \frac{1}{6}+\varepsilon & \frac{1}{6}-\varepsilon & \frac{1}{6}-\varepsilon & \frac{1}{6} \end{pmatrix}$$

x_1	x_2	R	outcome
1	2	$\frac{\frac{1}{6}\times\frac{1}{6}}{\left(\frac{1}{6}+\varepsilon\right)\times\frac{1}{6}}$	1
1	3	$\frac{\frac{1}{6}\times\frac{1}{6}}{\left(\frac{1}{6}+\varepsilon\right)\times\left(\frac{1}{6}+\varepsilon\right)}$	1
2	5	$\frac{\frac{1}{6}\times\frac{1}{6}}{\left(\frac{1}{6}-\varepsilon\right)\times\frac{1}{6}}$	0

$$\text{output } 1 \iff \frac{1}{(1+6\varepsilon)^{q_1+q_3}(1-6\varepsilon)^{q_4+q_5}} \leq 1 \overset{\sim}{\iff} q_4+q_5 \leq q_1+q_3$$

Example iii: Uniform over Different Supports

$$P_0 = \text{uniform} \qquad P_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \end{pmatrix}$$

x_1	x_2	R	outcome
1	2	$\frac{\frac{1}{5}\times\frac{1}{5}}{\frac{1}{4}\times\frac{1}{4}}$	1
1	3	$\frac{\frac{1}{5}\times\frac{1}{5}}{\frac{1}{4}\times\frac{1}{4}}$	1
2	5	$\frac{\frac{1}{5}\times\frac{1}{5}}{\frac{1}{4}\times 0}$	0

$$\text{output } 1 \Longleftrightarrow q_5 = 0$$

Example iv: Normal of Same Standard Deviation

$$P_0 = \mathcal{N}(\mu, \sigma) \qquad P_1 = \mathcal{N}(\mu', \sigma) \qquad \mu < \mu'$$

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

$$P_b \text{ : sum of } n \text{ instances of Ber}(p_b)$$

with $p_0 \approx p_1$ and $p_0 < p_1$
we approximate P_b to $\mathcal{N}(\mu_b, \sigma_b)$ with

$$\mu_b = np_b \qquad \sigma_b = \sqrt{np_b(1 - p_b)}$$

and $\sigma_0 \approx \sigma_1$
so

$$R \leq 1 \Longleftrightarrow \frac{x}{n} \geq \frac{p_0 + p_1}{2}$$

Problem

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}} &= d(P_0^{\otimes q}, P_1^{\otimes q}) \\ &= \frac{1}{2} \sum_{x_1,\ldots,x_q \in \mathcal{Z}} \left| \Pr_{P_1}[x_1] \cdots \Pr_{P_1}[x_q] - \Pr_{P_0}[x_1] \cdots \Pr_{P_0}[x_q] \right|\end{aligned}$$

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

$$|aa' - bb'| \le |a - b|\frac{a' + b'}{2} + |a' - b'|\frac{a + b}{2}$$

so

$$\begin{aligned}
& \frac{1}{2}\sum_{x_1,x_2} |P_1(x_1)Q_1(x_2) - P_0(x_1)Q_0(x_2)| \\
\le\ & \frac{1}{2}\sum_{x_1} |P_1(x_1) - P_0(x_1)| \sum_{x_2} \frac{Q_1(x_2) + Q_0(x_2)}{2} + \\
& \frac{1}{2}\sum_{x_2} |Q_1(x_2) - Q_0(x_2)| \sum_{x_1} \frac{P_1(x_1) + P_0(x_1)}{2} \\
=\ & \frac{1}{2}\sum_{x_1,x_2} |P_1(x_1) - P_0(x_1)| + \frac{1}{2}\sum_{x_1,x_2} |Q_1(x_2) - Q_0(x_2)| \\
=\ & d(P_0, P_1) + d(Q_0, Q_1)
\end{aligned}$$

Definitions

- **Kullback-Leibler divergence**

$$D_{\mathrm{KL}}(P_0\|P_1) = \sum_{x\in\mathrm{Supp}(P_0)} P_0(x)\log\frac{P_0(x)}{P_1(x)}$$

always non-negative, 0 iff $P_0 = P_1$
infinite iff $\mathrm{Supp}(P_0) \not\subseteq \mathrm{Supp}(P_1)$

WARNING: $\log$ are in basis 2!

- **Neyman divergence**

$$D_{\mathrm{N}}(P_0\|P_1) = \sum_{x\in\mathrm{Supp}(P_0)\cup\mathrm{Supp}(P_1)} \frac{(P_0(x)-P_1(x))^2}{P_1(x)}$$

always non-negative, 0 iff $P_0 = P_1$
infinite iff $\mathrm{Supp}(P_0) \not\subseteq \mathrm{Supp}(P_1)$

Better Bound

"for $q \ll 1/D_{\mathrm{N}}(P_0 \| P_1)$ the advantage must be $\ll 1$"

Theorem

For any q:

$$d(P_0^{\otimes q}, P_1^{\otimes q}) \leq \sqrt{\frac{q}{2} D_{\mathrm{N}}(P_0 \| P_1)}$$

Proof.

- $d(P_0^{\otimes q}, P_1^{\otimes q}) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P_0^{\otimes q} \| P_1^{\otimes q})}$ (Pinsker Inequality)
- $D_{\mathrm{KL}}(P_0^{\otimes q} \| P_1^{\otimes q}) = q D_{\mathrm{KL}}(P_0 \| P_1)$ (additivity of KL)
- $D_{\mathrm{KL}}(P_0 \| P_1) \leq D_{\mathrm{N}}(P_0 \| P_1)$ [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.

- $d(P^{\otimes q}, U^{\otimes q}) \leq \sqrt{\frac{q}{2} D_{\mathrm{N}}(P \| U)}$ (previous result)
- $D_{\mathrm{N}}(P \| U) = N \| P - U \|_2^2$ (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

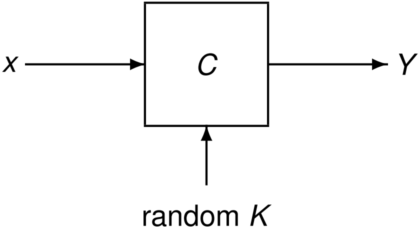
3. Cryptanalysis (Public-Key)

4. The Power of Interaction

5. **Cryptanalysis (Conventional)**
   - Block Ciphers
   - Differential Cryptanalysis
   - Linear Cryptanalysis
   - Hypothesis Testing in Cryptography
   - Decorrelation

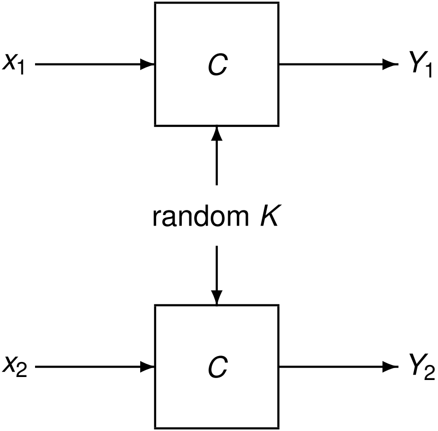
6. Proving Security

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.
$\Pr[Y_1 = y_1 \text{ and } Y_2 = y_2] = \frac{1}{2^\ell(2^\ell - 1)}$ for $y_1 \neq y_2$

n-wise Independence

Definition

For any pairwise different $x_1, \ldots, x_n$, the random variables $Y_i = C_K(x_i)$, $i = 1, \ldots, n$, defined by a random K are uniform and *nearly* independent if for any $y_1, \ldots, y_n$ of pairwise different values we have

$$\Pr[Y_1 = y_1, \ldots, Y_n = y_n] = \frac{1}{2^\ell (2^\ell - 1) \cdots (2^\ell - n + 1)}$$

- The perfect cipher is nearly n-wise independent
- The view from the adversary is $((x_1, y_1), \ldots, (x_n, y_n))$
- Real ciphers should look like n-wise independent so that adversaries limited to n samples get no information

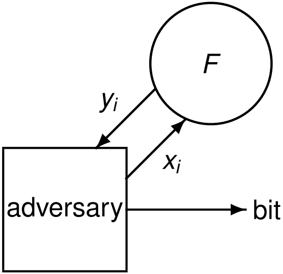
Goal of Block Cipher Designs

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

$\rightarrow$ uniform distribution and almost n-wise independence

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

- $F : A \to A$ 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

- $F : A \to B$ is a PRF or a truly random function F^*

  Hypothesis H_0: $F = F^*$ (uniformly distributes function)

  Hypothesis H_1: $F = F_K$ with K uniformly distributed

- $F : A \to A$ 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 $F : A \to B$ defined by K:

- $A = \{0, 1, 2\}$ and $B = \{0, 1\}$
- $F(x) = (K.x^2 + K.\lfloor \frac{K+x}{2} \rfloor + x + 1) \bmod 2$ 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

	$y = 0$	$y = 1$
$x = 0$	$1/4$	$3/4$
$x = 1$	$3/4$	$1/4$
$x = 2$	$1/4$	$3/4$

$$[F]^1 = \begin{pmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix} \quad , \quad [F^*]^1 = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

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

$$[F]^2 = \begin{pmatrix} 1/4 & 0 & 0 & 3/4 \\ 0 & 3/4 & 1/4 & 0 \\ 0 & 1/4 & 1/4 & 1/2 \\ 0 & 1/4 & 3/4 & 0 \\ 3/4 & 0 & 0 & 1/4 \\ 1/4 & 0 & 1/2 & 1/4 \\ 0 & 1/4 & 1/4 & 1/2 \\ 1/4 & 1/2 & 0 & 1/4 \\ 1/4 & 0 & 0 & 3/4 \end{pmatrix} \quad , \quad [F^*]^2 = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/2 & 0 & 0 & 1/2 \end{pmatrix}$$

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 $((x_1,\ldots,x_q),(y_1,\ldots,y_q))$-entry is

$$[F]^q_{(x_1,\ldots,x_q),(y_1,\ldots,y_q)} = \Pr[F(x_1)=y_1,\ldots,F(x_q)=y_q].$$

- 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

  $$\mathrm{Dec}^q(F) = D([F]^q,[F^*]^q).$$

  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

$$\begin{aligned} \mathrm{Dec}^2(F) &= |||[F]^2 - [F^*]^2|||_\infty \\ &= \max(0.5, 1, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5) \\ &= 1 \end{aligned}$$

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*

$$\mathrm{Adv}(F,G) = \frac{1}{2}|||[F]^q - [G]^q|||_\infty.$$

Proof. W.l.o.g. the best non-adaptive distinguisher is deterministic and asks the same question $x = (x_1, \ldots, x_q)$.

- the response $Y = (Y_1, \ldots, Y_q)$ 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

$$\mathrm{BestAdv}_q^{\mathrm{n.a.}}(F, F^*) = \frac{1}{2}|||[F]^q - [F^*]^q|||_\infty = \frac{1}{2}\mathrm{Dec}^q_{|||\cdot|||_\infty}(F)$$

A-Norm

Definition

Given a matrix M, we define

$$\|M\|_a = \max_{x_1} \sum_{y_1} \cdots \max_{x_q} \sum_{y_q} |M_{(x_1,\ldots,x_q),(y_1,\ldots,y_q)}|.$$

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*

$$\mathrm{Adv}(F,G) = \frac{1}{2}\|[F]^q - [G]^q\|_a.$$

Corollary

$$\mathrm{BestAdv}_q(F,F^*) = \frac{1}{2}|||[F]^q - [F^*]^q|||_a = \frac{1}{2}\mathrm{Dec}^q_{\|\cdot\|_a}(F)$$

Multiplicativity of Decorrelation

Theorem

If C_1 and C_2 are two independent random permutations over a set A,

$$\mathrm{Dec}^q(C_2 \circ C_1) \leq \mathrm{Dec}^q(C_1)\mathrm{Dec}^q(C_2)$$

when Dec *is defined from a matrix norm.*

so,

$$2\mathrm{BestAdv}_q(C_2 \circ C_1, C^*) \leq (2\mathrm{BestAdv}_q(C_1, C^*)) \times (2\mathrm{BestAdv}_q(C_2, C^*))$$

Corollary

If $C_1, \ldots, C_r$ are iid random permutations,

$$\mathrm{BestAdv}_q(C_r \circ \cdots \circ C_1, C^*) \leq \frac{1}{2}(2\mathrm{BestAdv}_q(C_1, C^*))^r$$

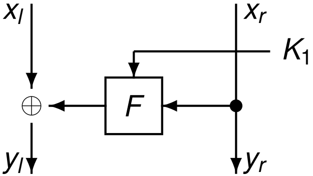
Proof

- 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 $([C_1]^q - [C^*]^q) \times ([C_2]^q - [C^*]^q) = [C_2 \circ C_1]^q - [C^*]^q$ 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 $> \frac{1}{2}$)

$\Psi(F^{K_1})$

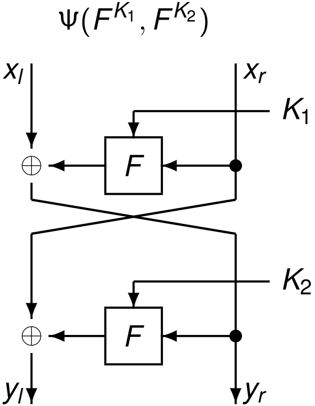


$$x_r = y_r$$

so, we can make a distinguisher with advantage $1 - 2^{-\frac{\ell}{2}}$:

```
1: pick $x$ random
2: get $y = C(x)$
3: output $1_{x_r = y_r}$
```

2-Round Feistel Scheme is no Good (BestAdv $> \frac{1}{2}$)



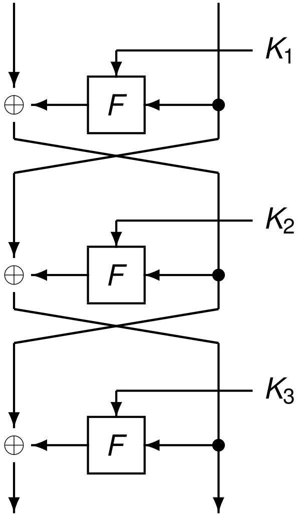
$$x_r = x'_r \Longrightarrow x_l \oplus y_r = x'_l \oplus y'_r$$

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

$\Psi(F^{K_1}, F^{K_2}, F^{K_3})$



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

$$\begin{aligned} \mathrm{BestAdv}_q(\Psi(F_1^*, F_2^*, F_3^*), F^*) &\leq q^2.2^{-\frac{\ell}{2}} \\ \mathrm{BestAdv}_q(\Psi(F_1^*, F_2^*, F_3^*), C^*) &\leq q^2.2^{-\frac{\ell}{2}} \end{aligned}$$

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 $F : \mathcal{M}_1 \to \mathcal{M}_2$ be a random function. We let $\mathcal{X}$ be the subset of $\mathcal{M}_1^q$ of all $(x_1, \ldots, x_q)$ with pairwise different entries. We let $F^* : \mathcal{M}_1 \to \mathcal{M}_2$ be a uniformly distributed random function. We assume there exists a subset $\mathcal{Y} \subseteq \mathcal{M}_2^q$ and two positive numbers ϵ_1 and ϵ_2 such that

- $\frac{\#\mathcal{Y}}{\#\mathcal{M}_2^q} \geq 1 - \epsilon_1$

this is $[F^*]^q_{x,y}$

- $\forall x \in \mathcal{X} \quad \forall y \in \mathcal{Y} \quad [F]^q_{x,y} \geq \frac{1}{\#\mathcal{M}_2^q}(1 - \epsilon_2)$.

Then we have $\mathsf{BestAdv}_q(F, F^*) \leq \epsilon_1 + \epsilon_2$.

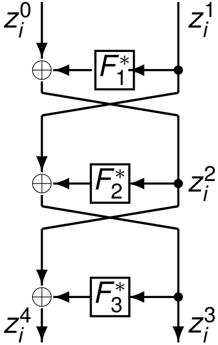
"if $[F]^q_{x,y} \approx [F^*]^q_{x,y}$ for almost all y's, then $\mathsf{BestAdv}_q(F, F^*)$ is small"

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 $i = 1, \ldots, q$.
- Event E^2: all z_i^2s are pairwise different (depends on F_1^* only).
- $\mathcal{Y} = \left\{(y_1, \ldots, y_q); \forall i < j \quad z_i^3 \neq z_j^3\right\}$.



We have $[F]^q_{x,y} = \Pr[E]$

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 $[F]^q_{x,y}$. We have
$$[F]^q_{x,y} = \Pr[E] \geq \Pr[E \wedge E^2] = \Pr[E|E^2] \Pr[E^2].$$
For computing $\Pr[E|E^2]$ we know that z_i^3s are pairwise different, as for the z_i^2s. 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 $\mathsf{BestAdv}_q(F, F^*) \leq q(q-1)2^{-\frac{\ell}{2}}$.
It remains to show $\mathsf{BestAdv}_q(F^*, C^*) \leq q 2^{-\frac{\ell}{2}}$ (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 $\mathrm{BestAdv}_q(F,C) \leq \frac{q(q-1)}{2}2^{-\ell}$.

So, $\mathrm{BestAdv}_q(F,C) \leq \min(q^2 2^{-\ell},1) \leq \min(q2^{-\frac{\ell}{2}},1)$.
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 ith query.
Conditioned to the event E : no $F(x_i)$ collide, the distribution of $(F(x_1),\ldots,F(x_q))|E$ and $(C(x_1),\ldots,C(x_q))$ 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, $\Pr[\neg E] \leq \sum_{1\leq i<j\leq q}^{q} \Pr[F(x_i) = F(x_j)] = \frac{q(q-1)}{2}2^{-\ell}$. □

$\Pr[A]=\Pr[A,E]+\Pr[A,\neg E]\leq\Pr[A|E]\Pr[E]+\Pr[\neg E]\leq\Pr[A|E]+\Pr[\neg E]$

Proof of Lemma — i

- We use the characterization of $\mathrm{Dec}^q_{\|\cdot\|_a}$ 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 $y = (y_1, \ldots, y_q)$. We let x_i denotes the ith query defined by y and $x = (x_1, \ldots, x_q)$ be defined by y. We let A be the set of all y for which $\mathcal{A}$ accepts. We have

$$\mathrm{Adv}_{\mathcal{A}}(F, F^*) = \Pr[\mathcal{A}(F^*)] - \Pr[\mathcal{A}(F)] = \sum_{y \in A} \left([F^*]^q_{x,y} - [F]^q_{x,y}\right).$$

Since $[F^*]^q_{x,y} - [F]^q_{x,y} \leq \epsilon_2 [F^*]^q_{x,y}$ in $\mathcal{Y}$, we have

$$\mathrm{Adv}_{\mathcal{A}}(F, F^*) \leq \sum_{\substack{y \in A \\ y \in \mathcal{Y}}} \epsilon_2 [F^*]^q_{x,y} + \sum_{\substack{y \in A \\ y \notin \mathcal{Y}}} [F^*]^q_{x,y}.$$

Proof of Lemma — ii

$$\mathsf{Adv}_{\mathcal{A}}(F,F^*) \leq \sum_{\substack{y\in A\\ y\in\mathcal{Y}}} \epsilon_2 [F^*]^q_{x,y} + \sum_{\substack{y\in A\\ y\notin\mathcal{Y}}} [F^*]^q_{x,y}.$$

- The first sum is upper bounded by ϵ_2:

$$\sum_{\substack{y\in A\\ y\in\mathcal{Y}}} [F^*]^q_{x,y} \leq \sum_{y} [F^*]^q_{x,y} = 1$$

- For the second sum, we recall that all x_is are pairwise different, so

$$\sum_{\substack{y\in A\\ y\notin\mathcal{Y}}} [F^*]^q_{x,y} = \sum_{\substack{y\in A\\ y\notin\mathcal{Y}}} \frac{1}{\#\mathcal{M}_2^q} \leq \sum_{y\notin\mathcal{Y}} \frac{1}{\#\mathcal{M}_2^q} = \frac{\#\mathcal{M}_2^q - |\mathcal{Y}|}{\#\mathcal{M}_2^q} \leq \varepsilon_1$$

so $\mathsf{Adv}_{\mathcal{A}}(F,F^*) \leq \varepsilon_1 + \epsilon_2$. □

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 $\mathrm{BestAdv}_q(F_i, F^*) \leq \varepsilon$. We have

$$\mathrm{BestAdv}_q(\Psi(F_1, F_2, F_3), C^*) \leq q^2.2^{-\frac{\ell}{2}} + 3\varepsilon$$

where C^* is a uniformly distributed random permutation.

Proof.

$$\begin{aligned}
& \mathrm{BestAdv}_q(\Psi(F_1,F_2,F_3),C^*) \\
\leq\ & \mathrm{BestAdv}_q(\Psi(F_1,F_2,F_3),\Psi(F_1,F_2,F_3^*)) + \mathrm{BestAdv}_q(\Psi(F_1,F_2,F_3^*),\Psi(F_1,F_2^*,F_3^*)) + \\
& \mathrm{BestAdv}_q(\Psi(F_1,F_2^*,F_3^*),\Psi(F_1^*,F_2^*,F_3^*)) + \mathrm{BestAdv}_q(\Psi(F_1^*,F_2^*,F_3^*),C^*) \\
\leq\ & \underbrace{\mathrm{BestAdv}_q(F_3,F_3^*)}_{\leq\varepsilon} + \underbrace{\mathrm{BestAdv}_q(F_2,F_2^*)}_{\leq\varepsilon} + \underbrace{\mathrm{BestAdv}_q(F_1,F_1^*)}_{\leq\varepsilon} + \underbrace{\mathrm{BestAdv}_q(\Psi(F_1^*,F_2^*,F_3^*),C^*)}_{\leq q^2.2^{-\frac{\ell}{2}} \quad [LR]}
\end{aligned}$$

□

Iterating

Theorem

Let $F_1, \ldots, F_{3r}$ be $3r$ independent random functions on $\{0,1\}^{\frac{\ell}{2}}$ such that $\mathrm{BestAdv}_q(F_i, F^*) \leq \varepsilon$. We have

$$\mathrm{BestAdv}_q(\Psi(F_1, \ldots, F_{3r}), C^*) \leq \frac{1}{2}\left(2q^2.2^{-\frac{\ell}{2}} + 6\varepsilon\right)^r$$

where C^* is a uniformly distributed random permutation.

Provable Security on Feistel Schemes

$$\mathrm{BestAdv}_q(\Psi(F_1,\ldots,F_{3r}),C^*) \le \frac{1}{2}\left(2q^2.2^{-\frac{\ell}{2}}+6\varepsilon\right)^r$$

- 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}
$\frac{1}{2}\left(2q^2.2^{-\frac{\ell}{2}}\right)^r$	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 $q = 2$

$E\left(\mathrm{DP}^{C^*}(a,b)\right)$

Theorem

Given a random C on $\{0,1\}^\ell$ and $a, b \neq 0$ we have

$$
\begin{aligned}
E\left(\mathrm{DP}^{C}(a,b)\right) &\leq \frac{1}{2^\ell - 1} + \frac{1}{2}|||[C]^2 - [C^*]^2|||_\infty \\
E\left(\mathrm{LP}^{C}(a,b)\right) &\leq \frac{1}{2^\ell - 1} + 4|||[C]^2 - [C^*]^2|||_\infty
\end{aligned}
$$

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
  final exam 2021–22 ex2 (optimal LP)
  final exam 2022–23 ex2 (finding heavy differentials)
- 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)
  final exam 2018–19 ex1 (number of samples)
  final exam 2019–20 ex2 (advantage amplification)
  final exam 2021–22 ex2 (Lai-Massey)
  final exam 2021–22 ex3 (mod *p* PRNG)

Train Yourself

- 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
- Even-Mansour cipher: final exam 2022–23 ex1
- a simple PRF: final exam 2023–24 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

5 Cryptanalysis (Conventional)

6 **Proving Security**
- The Random Oracle Model
- Hybrid ElGamal
- The Fujisaki-Okamoto Transform
- The Generic Group Model

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
  $\rightarrow$ 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 ℓ bits we can assume that H yields ℓ-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 form the correct one

Full-Domain Hash (FDH) Signature

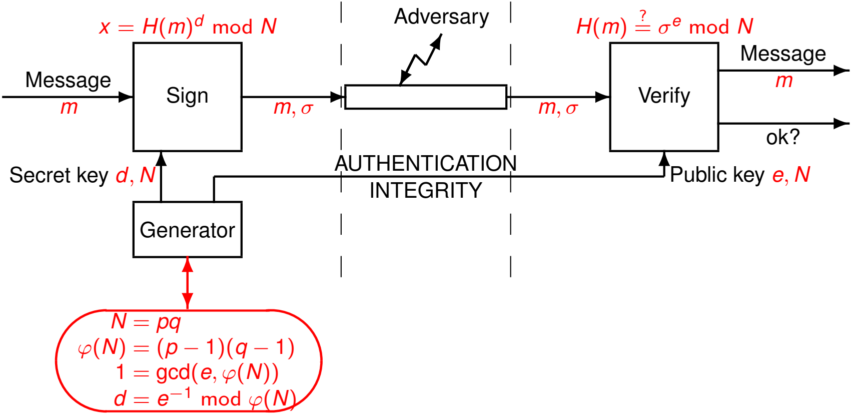
- consider the RSA homomorphic trapdoor permutation over $\mathbf{Z}_N^*$
- let H be a random oracle hashing onto $\mathbf{Z}_N^*$
- $\mathsf{Sign}_{d,N}(m) = (H(m))^d \bmod N$
- $\mathsf{Verify}_{e,N}(m,\sigma) \iff \sigma^e \bmod N = H(m)$

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 ε 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.

$(\exp(x) = 2.71\ldots^x)$

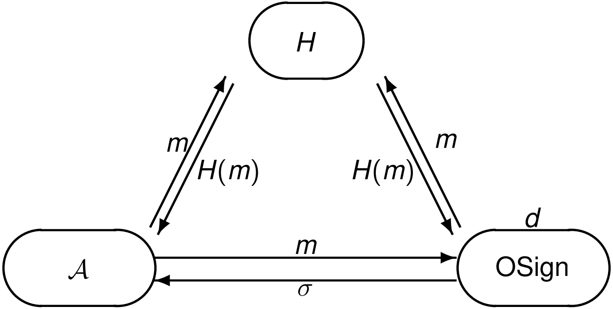
Full-Domain Hash (FDH) Signature



Proof — i

Let $\varepsilon = \Pr[\mathcal{A} \text{ wins}]$.

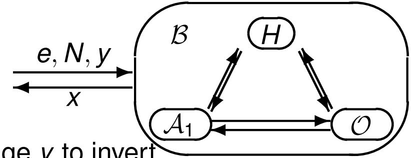
- the EF-CMA game selects pk and sk and gives pk to $\mathcal{A}$, then OSign answers to any signature request
- example: if m is a signature query, OSign queries m to H, gets h, and answers $h^d \bmod N$ to $\mathcal{A}$



Proof — ii

- define $\mathcal{A}_1$ as follows
  - simulate $\mathcal{A}$ until $\mathcal{A}$ yields its final (m, σ) forgery
  - if m was queried to OSign, abort
    $\rightarrow$ m not queried to H by OSign
  - 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, σ)
- we obtain: a new EF-CMA adversary $\mathcal{A}_1$ with similar complexity and same success probability ε, who either aborts or yields a final result (m, σ) to win, 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 OSign
  - upon a query m to H, if m was queried before, give the same answer; otherwise, pick $r \in_U \mathbf{Z}_N^*$, 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, σ), if query m to H produced the answer of type $yr^e \bmod N$, yield $\sigma/r \bmod N$; otherwise, abort
    → if $H(m) = yr^e \bmod N$, 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 $yr^e \bmod N$ 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

$$\frac{1}{q_S+1}\left(1-\frac{1}{q_S+1}\right)^{q_S}\varepsilon \geq \frac{\exp(-1)}{q_S+1}\varepsilon$$

□

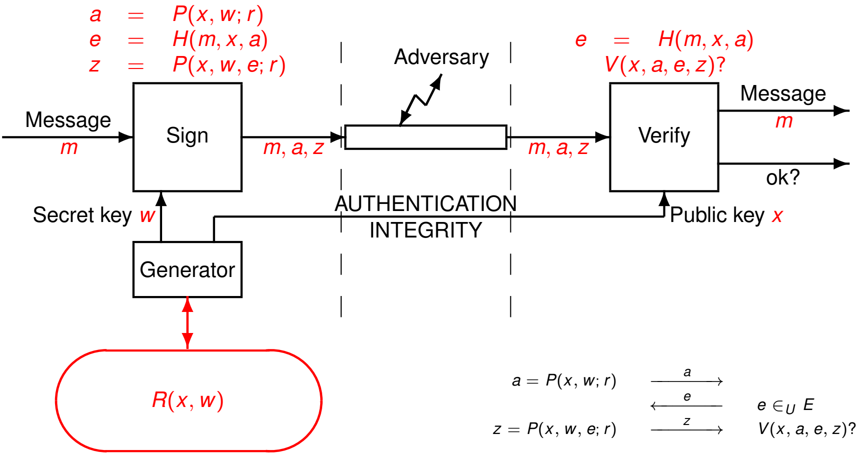
Fiat-Shamir Signature Paradigm

Signature from a Σ-Protocol

- Σ-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 $a = \mathcal{P}(x, w; r)$,
  $e = H(\text{message}\|x\|a) \in E$, $z = \mathcal{P}(x, w, e; r)$
  signature is (a, z)
- verification: $V(x, a, H(\text{message}\|x\|a), z)$

note: the x missing in $e = H(\text{message}\|x\|a)$ is an original mistake!

Fiat-Shamir Signature Paradigm



$$\begin{array}{rcl} a = P(x,w;r) & \xrightarrow{a} & \\ & \xleftarrow{e} & e \in_U E \\ z = P(x,w,e;r) & \xrightarrow{z} & V(x,a,e,z)? \end{array}$$

Fiat-Shamir Signature Paradigm in ROM

Theorem

Given a relation R s.t. it is hard to find witnesses and a Σ-protocol for its language s.t. $1/\#E = \mathrm{negl}$, 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

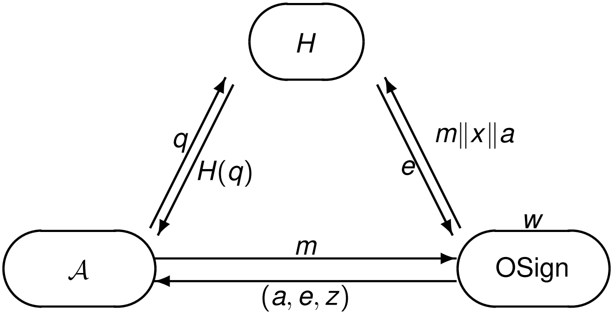
Given a relation R s.t. it is hard to find witnesses and a Σ-protocol for its language s.t. $1/\#E = \mathsf{negl}$, we consider the signature scheme obtained by the Fiat-Shamir construction ***using a random oracle****.*
There is a compiler which can transform an adversary $\mathcal{A}$ succeeding the 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

$$\Pr[\mathcal{A}' \text{ wins}] = \Pr[\mathcal{A} \text{ wins}] - \mathsf{negl}$$

Proof of Lemma — i

Let $\varepsilon = \Pr[\mathcal{A} \text{ wins}]$.

- modify: OSign and $\mathcal{A}$ output e in signatures
- the EF-CMA game selects x and w and gives x to $\mathcal{A}$, then OSign answers to any signature query
- example: if m is a signature query, the challenger picks r, computes $a = \mathcal{P}(x, w; r)$, queries $m\|x\|a$ 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 OSign, abort
      $\rightarrow$ no query of form $m\|x\|a'$ from OSign to H
    - if $m\|x\|a$ was not queried to H by $\mathcal{A}$, query it
    - if $\neg V(x, a, e, z)$ or $e \neq H(m\|x\|a)$, abort
    - yield (m, a, e, z)
- we obtain: a new EF-CMA adversary $\mathcal{A}_1$ with similar complexity and success probability ε, who either aborts or yields a final result (m, a, e, z) and wins, and who always queries $(m\|x\|a)$ to H

Proof of Lemma — iii

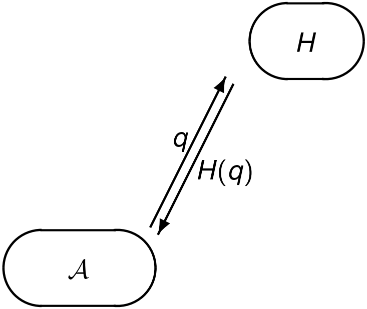
- define $\mathcal{A}_2$ as follows
  - simulate $\mathcal{A}_1$ and make a list of all $(q, H(q))$ queries (queries $m\|x\|a \mapsto e$ by OSign are deduced from x, m, and (a, e, z))
  - if adversary tries to repeat a query which was done before, just take its answer from the list and avoid the query
  - if OSign does a query which was done before (let ε' 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 repeats any query
- ε' is negligible:
  $\#$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 commit 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

Proof of Lemma — iv

- define $\mathcal{A}'$ as follows
  - simulate $\mathcal{A}_2$ until a query m to OSign is made
  - upon query m, $\mathcal{A}'$ picks r, e, and computes $(a, e, z) = \mathcal{S}(x, e; r)$
    if $m \| a$ has been queried before, the simulation fails
    otherwise, the H table is augmented with $m \| x \| a \mapsto e$
- we obtain: an EF-0MA adversary $\mathcal{A}'$ with similar complexity and success probability $\varepsilon - \varepsilon'$

□

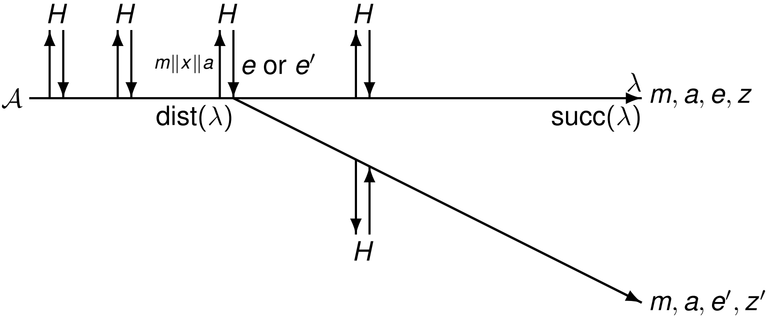
Proof of Theorem — i



define $\mathcal{B}$ as follows

- simulate $\mathcal{A}$ with initial x, 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 $m\|x\|a$ 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 ν corresponds to a query, each leaf λ corresponds to a termination)

- we consider the random descent following the H simulation which leads to a random leaf X
- let succ(λ) be true if λ yields (m, a, e, z) and false otherwise
- if $\lambda \to (m, a, e, z)$, let dist(λ) be the ancestor of λ who made to the $m\|x\|a$ query, otherwise, let $\mathrm{dist}(\lambda) = \lambda$
- we have $\Pr[\mathrm{succ}(X)] = \varepsilon$ and $E(\mathrm{depth}(X)) = \mathrm{poly}$
- let visit(ν) the event that X has ν as an ancestor
  let $f(\nu) = \Pr[\mathrm{succ}(X), \mathrm{dist}(X) = \nu | \mathrm{visit}(\nu)]$

we obtain: a witness extractor $\mathcal{B}$ with similar complexity who succeeds if succ(X), $e \neq e'$, and the second run succeeds on X' such that $\mathrm{dist}(X') = \mathrm{dist}(X)$.

since $\Pr[e = e'] = \mathrm{negl}$, the success probability is greater than $E(f(\mathrm{dist}(X))) - \mathrm{negl}$

Proof of Theorem — iv

recap:

- we have a tree with a predicate succ on leaves and

$$\text{dist} : \lambda \mapsto \begin{cases} \text{one ancestor if succ}(\lambda) \\ \lambda \text{ otherwise} \end{cases}$$

- 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 ν is an ancestor of X
- if $E(f(\text{dist}(X)))$ is negligible, so is ε'

we conclude using the Forking Lemma □

Forking Lemma

Lemma (Forking Lemma)

- *We consider a finite tree and a mapping* dist *which maps any leaf λ to one of its ancestors* $\mathrm{dist}(\lambda)$*. We call it a distinguished ancestor.*
- *We assume we are given a distribution which defines a random leaf X. We let* $\mathrm{visit}(\nu)$ *be the event that the descent goes through ν, i.e. that ν is an ancestor of λ.*
- *We let* $\mathrm{succ}(\lambda)$ *be true iff* $\mathrm{dist}(\lambda) \neq \lambda$*. When it occurs we say that λ is successful.*
- *We let* $p = \Pr[\mathrm{succ}(X)]$, $\bar{d} = E(\mathrm{depth}(X))$, *and* $f(\nu) = \Pr[\mathrm{succ}(X) \text{ and } \mathrm{dist}(X) = \nu | \mathrm{visit}(\nu)]$.

We have

$$\Pr\left[f(\mathrm{dist}(X)) > \frac{p}{2\bar{d}}\middle|\mathrm{succ}(X)\right] \geq \frac{1}{2}$$

Consequence

$$\begin{aligned} E(f(\mathrm{dist}(X))) &= \int_0^1 \Pr[f(\mathrm{dist}(X)) \geq t]\, dt \\ &\geq \int_0^1 \Pr[f(\mathrm{dist}(X)) \geq t, \mathrm{succ}(X)]\, dt \\ &= p \int_0^1 \Pr[f(\mathrm{dist}(X)) \geq t | \mathrm{succ}(X)]\, dt \\ &\geq p \int_0^1 \frac{1}{2} \cdot 1_{t \leq \frac{p}{\bar{d}}}\, dt \\ &= \frac{p^2}{4\bar{d}} \end{aligned}$$

so, $p \leq \sqrt{4\bar{d}E(f(\mathrm{dist}(X)))}$
If $E(f(\mathrm{dist}(X)))$ is negligible, p is negligible as well, which completes the proof of the Theorem.

Proof of Forking Lemma

- we have $\Pr[\mathsf{dist}(X) = \nu | \mathsf{succ}(X)] = f(\nu)\frac{\Pr[\mathsf{visit}(\nu)]}{p}$
- let Bad be the set of ν's s.t. $f(\nu) \leq \frac{p}{2\bar{d}}$
  we have

$$\begin{aligned}
\Pr[\mathsf{dist}(X) \in \mathsf{Bad} | \mathsf{succ}(X)] &= \sum_{\nu \in \mathsf{Bad}} f(\nu)\frac{\Pr[\mathsf{visit}(\nu)]}{p} \\
&\leq \frac{\sum_\nu \Pr[\mathsf{visit}(\nu)]}{2\bar{d}} \\
&\leq \frac{1}{2}
\end{aligned}$$

- so, $\Pr[\mathsf{dist}(X) \notin \mathsf{Bad} | \mathsf{succ}(X)] \geq \frac{1}{2}$

□

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 instanciation of the random oracle

- security in this model is 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

$$\mathsf{Sign}^0_{d,N}(m) = (H(m))^d \bmod N$$

- construct another ROM-secure digital signature scheme:
  - message semantics: interpret m as an algorithm implementing a partial function h_m within a bounded time τ
  - $\mathsf{Sign}_{d,N}(m)$: pick r; if $H(r) = h_m(r)$ then output d; otherwise output $\mathsf{Sign}^0_{d,N}(m)$
  - $\mathsf{Verify}_{e,N}(m, \sigma) \Longleftrightarrow m^{e\sigma} \bmod N = m$ 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
   - The Generic Group Model

Hybrid ElGamal Cryptosystem using the Leftover Hash Lemma

take a group with a generator g of order q

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

message space: $M \in \{0,1\}^m$

encryption: $\mathsf{Enc}_y(M; r, n) = (g^r, M \oplus h_n(y^r), n)$

decryption: $\mathsf{Dec}_x(u, v, n) = v \oplus h_n(u^x)$

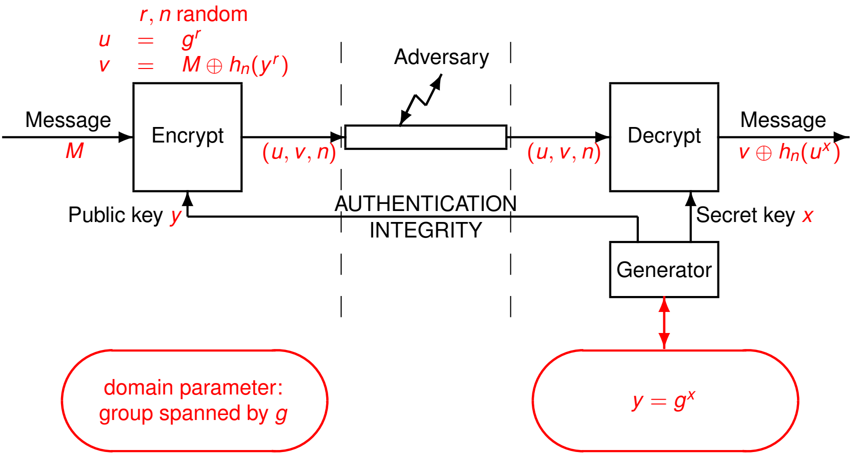
where $(h_n)_n$ is a family of universal hash functions from $G \supseteq \langle g \rangle$ to $\{0,1\}^m$

leftover hash Lemma: if $\varepsilon \geq 2^{\frac{m}{2}} / \sqrt{q}$ then $(h_n(g^s), n)$ is ε-indistinguishable from (u, n) where $u \in_U \{0,1\}^m$

Theorem

If $2^{\frac{m}{2}} / \sqrt{q} = \mathrm{negl}$ *and under the DDH assumption, the above is IND-CPA-secure.*

Hybrid ElGamal Cryptosystem



Leftover Hash Lemma

example: $H_\infty(g^s) = \log_2 q$

- **min-entropy**:

$$H_\infty(X) = -\log_2 \max_x \Pr[X = x]$$

- **universal hash 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.

Proof

let $P_0 = (h_N(X), N)$, $P_1 = (U, N)$, compute the Euclidean distance:

$$\begin{aligned}
\|P_1 - P_0\|_2^2 &= \sum_{k,n}\left(\Pr_{X,N}[h_n(X) = k, N = n] - \frac{1}{2^m \#\mathcal{N}}\right)^2 \\
&= \left(\sum_{k,n}\Pr_{X,N}[h_n(X) = k, N = n]^2\right) - \frac{1}{2^m \#\mathcal{N}} \\
&= \Pr_{X,X',N,N'}[h_N(X) = h_{N'}(X'), N = N'] - \frac{1}{2^m \#\mathcal{N}} \\
&= \frac{1}{\#\mathcal{N}}\sum_{x,x'}\Pr[X = x, X' = x', h_N(x) = h_N(x')] - \frac{1}{2^m \#\mathcal{N}} \\
&= \frac{1-2^{-m}}{\#\mathcal{N}}\sum_{x}\Pr[X = x]^2 \quad \text{(split } x = x' \text{ and } x \neq x'\text{)} \\
&\leq \frac{1-2^{-m}}{\#\mathcal{N}}\max_x \Pr[x] \;\leq\; \frac{1-2^{-m}}{\#\mathcal{N}}2^{-H_\infty(X)} \;\leq\; \frac{1}{2^m \#\mathcal{N}}\varepsilon^2
\end{aligned}$$

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 $\Gamma_0^b$:
    run key generation and get $y$
    pick $\rho$ and set view $= (y;\rho)$
    run $\mathcal{A}(\text{view}) = (m_0, m_1)$
    pick $r$, $u \leftarrow g^r$
    pick $n$, $v \leftarrow m_b \oplus h_n(y^r)$
    set view $= (y, u, v, n; \rho)$
    run $\mathcal{A}(\text{view}) = b'$
    return $b'$
```

$\updownarrow$ bridge

```
game $\Gamma_1^b$:
    pick $x$, $y \leftarrow g^x$
    pick $r$, $u \leftarrow g^r$
    $X \leftarrow g^{xr}$                ▷ erase $x, r$
    pick $\rho$ and set view $= (y;\rho)$
    run $\mathcal{A}(\text{view}) = (m_0, m_1)$
    pick $n$, $v \leftarrow m_b \oplus h_n(X)$
    set view $= (y, u, v, n; \rho)$
    run $\mathcal{A}(\text{view}) = b'$
    return $b'$
```

$\overset{\text{DDH}}{\approx}$

DDH assumption in the group

```
game $\Gamma_2^b$:
    pick $x$, $y \leftarrow g^x$
    pick $r$, $u \leftarrow g^r$
    pick $s$, $X \leftarrow g^s$         ▷ erase $x, r, s$
    pick $\rho$ and set view $= (y;\rho)$
    run $\mathcal{A}(\text{view}) = (m_0, m_1)$
    pick $n$, $v \leftarrow m_b \oplus h_n(X)$
    set view $= (y, u, v, n; \rho)$
    run $\mathcal{A}(\text{view}) = b'$
    return $b'$
```

Transitions — ii

```
game $\Gamma_2^b$:
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $s$, $X \leftarrow g^s$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}(\text{view}) = (m_0, m_1)$
pick $n$, $v \leftarrow m_b \oplus h_n(X)$
set view $= (y, u, v, n; \rho)$
run $\mathcal{A}(\text{view}) = b'$
return $b'$
```

$\updownarrow$ bridge

```
game $\Gamma_3^b$:
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}(\text{view}) = (m_0, m_1)$
pick $s$, $X \leftarrow g^s$
pick $n$, $v_0 \leftarrow h_n(X)$   $\triangleright$ erase $s, X$
$v \leftarrow m_b \oplus v_0$
set view $= (y, u, v, n; \rho)$
run $\mathcal{A}(\text{view}) = b'$
return $b'$
```

leftover hash Lemma

$\overset{\text{lemma}}{=}$

```
game $\Gamma_4^b$:
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}(\text{view}) = (m_0, m_1)$
pick $U$
pick $n$, $v_0 \leftarrow U$   $\triangleright$ erase $U$
$v \leftarrow m_b \oplus v_0$
set view $= (y, u, v, n; \rho)$
run $\mathcal{A}(\text{view}) = b'$
return $b'$
```

Transitions — iii

game Γ_4^b:

```
pick $x, y \leftarrow g^x$
pick $r, u \leftarrow g^r$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}(\text{view}) = (m_0, m_1)$
pick $U$
pick $n, v_0 \leftarrow U$
$v \leftarrow m_b \oplus v_0$
set view $= (y, u, v, n; \rho)$
run $\mathcal{A}(\text{view}) = b'$
**return** $b'$
```

$\updownarrow$ bridge

game Γ_5^b:

```
pick $x, y \leftarrow g^x$
pick $r, u \leftarrow g^r$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}(\text{view}) = (m_0, m_1)$
pick $n$
pick $v_0$
$v \leftarrow m_b \oplus v_0$        ▷ erase $v_0$
set view $= (y, u, v, n; \rho)$
run $\mathcal{A}(\text{view}) = b'$
**return** $b'$
```

$\overset{\text{ind}}{=}$

v_0 and v uniform

game Γ_6^b:

```
pick $x, y \leftarrow g^x$
pick $r, u \leftarrow g^r$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}(\text{view}) = (m_0, m_1)$
pick $n$
pick $v$
set view $= (y, u, v, n; \rho)$
run $\mathcal{A}(\text{view}) = b'$
**return** $b'$
```

Transitions — iv

final step: Γ_6^0 and Γ_6^1 are identical!

$$\begin{array}{ccccccccccccc}
\Gamma_0^0 & \overset{\text{bridge}}{\frown} & \Gamma_1^0 & \overset{\text{DDH}}{\approx} & \Gamma_2^0 & \overset{\text{bridge}}{\frown} & \Gamma_3^0 & \overset{\text{lemma}}{\approx} & \Gamma_4^0 & \overset{\text{bridge}}{\frown} & \Gamma_5^0 & \overset{\text{domain}}{=} & \Gamma_6^0 \\
 & & & & & & & & & & & & \| \\
\Gamma_0^1 & \overset{\text{bridge}}{\frown} & \Gamma_1^1 & \overset{\text{DDH}}{\approx} & \Gamma_2^1 & \overset{\text{bridge}}{\frown} & \Gamma_3^1 & \overset{\text{lemma}}{\approx} & \Gamma_4^1 & \overset{\text{bridge}}{\frown} & \Gamma_5^1 & \overset{\text{domain}}{=} & \Gamma_6^1
\end{array}$$

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

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: $\mathsf{Enc}_y(M; r) = (g^r, M \oplus H(y^r))$

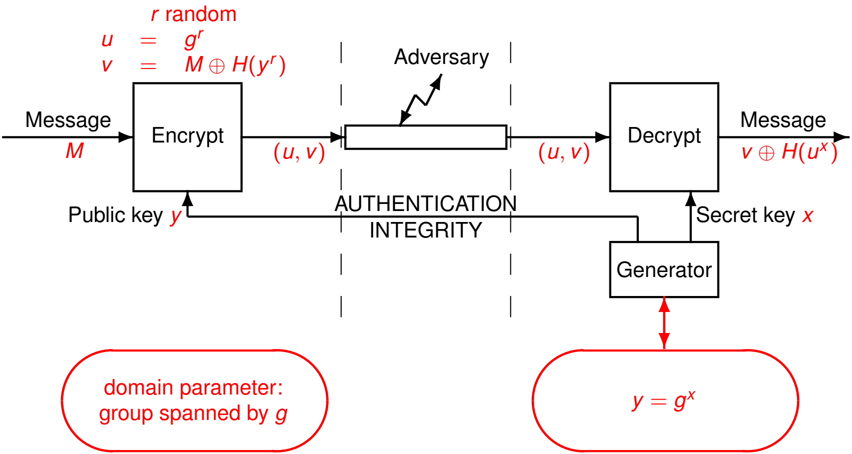
decryption: $\mathsf{Dec}_x(u, v) = v \oplus H(u^x)$

where H is a random oracle onto $\{0,1\}^m$

Theorem

Under the DDH assumption, the above is INDCPA-secure in the random oracle model.

Hybrid ElGamal Cryptosystem



Transitions — i

game Γ_0^b:

```
pick $H$
run key generation and get $y$
pick $\rho$ and set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
pick $r$, $u \leftarrow g^r$
$v \leftarrow m_b \oplus H(y^r)$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
return $b'$
```

$\updownarrow$ bridge

game Γ_1^b:

```
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
$X \leftarrow g^{xr}$
                                    ▷ erase $x, r$
pick $H, \rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
$v \leftarrow m_b \oplus H(X)$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
return $b'$
```

$\overset{\text{DDH}}{\approx}$

DDH assumption in the group

game Γ_2^b:

```
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $s$, $X \leftarrow g^s$
                                    ▷ erase $x, r, s$
pick $H, \rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
$v \leftarrow m_b \oplus H(X)$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
return $b'$
```

Transitions — ii

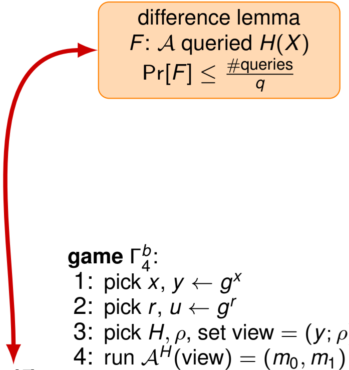
game Γ_2^b:

```
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $s$, $X \leftarrow g^s$
pick $H$, $\rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
$v \leftarrow m_b \oplus H(X)$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
return $b'$
```

$\updownarrow$ bridge

game Γ_3^b:

```
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $H$, $\rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
pick $s$, $X \leftarrow g^s$
$v_0 \leftarrow H(X)$
$v \leftarrow m_b \oplus v_0$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
return $b'$
```



difference lemma

F: $\mathcal{A}$ queried $H(X)$

$\Pr[F] \leq \frac{\#\text{queries}}{q}$

$\overset{\Pr[F]}{\approx}$

game Γ_4^b:

```
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $H$, $\rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
pick $s$, $X \leftarrow g^s$
pick $v_0$
$v \leftarrow m_b \oplus v_0$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
**return** 0 **if** $\mathcal{A}$ queried $H(X)$
return $b'$
```

Transitions — iii

game Γ_4^b:

```
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $H, \rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
pick $s$, $X \leftarrow g^s$
pick $v_0$, $v \leftarrow m_b \oplus v_0$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
**return** 0 **if** $\mathcal{A}$ queried $H(X)$
return $b'$
```

$\updownarrow$ bridge

v_0 and v uniform

game Γ_5^b:

```
pick $v_0$, $v \leftarrow m_b \oplus v_0$
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $H, \rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
pick $s$, $X \leftarrow g^s$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
**return** 0 **if** $\mathcal{A}$ queried $H(X)$
return $b'$
```

$\overset{\text{ind}}{=}$

game Γ_6^b:

```
pick $v$
pick $x$, $y \leftarrow g^x$
pick $r$, $u \leftarrow g^r$
pick $H, \rho$, set view $= (y; \rho)$
run $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
pick $s$, $X \leftarrow g^s$
set view $= (y, u, v; \rho)$
run $\mathcal{A}^H(\text{view}) = b'$
**return** 0 **if** $\mathcal{A}$ queried $H(X)$
return $b'$
```

Transitions — iv

final step: Γ_6^0 and Γ_6^1 are identical!

$$\begin{tikzcd}
\Gamma_0^0 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma_1^0 \arrow[r, phantom, "\overset{\text{DDH}}{\approx}"] & \Gamma_2^0 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma_3^0 \arrow[r, phantom, "\overset{\text{difference}}{\approx}"] & \Gamma_4^0 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma_5^0 \arrow[r, phantom, "\overset{\text{domain}}{=}"] & \Gamma_6^0 \arrow[d, phantom, "\|"] \\
\Gamma_0^1 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma_1^1 \arrow[r, phantom, "\overset{\text{DDH}}{\approx}"] & \Gamma_2^1 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma_3^1 \arrow[r, phantom, "\overset{\text{difference}}{\approx}"] & \Gamma_4^1 \arrow[r, phantom, "\overset{\text{bridge}}{\frown}"] & \Gamma_5^1 \arrow[r, phantom, "\overset{\text{domain}}{=}"] & \Gamma_6^1
\end{tikzcd}$$

so, $\Pr[\Gamma_0^0 = 1] - \Pr[\Gamma_0^1 = 1] \leq 2\mathsf{Adv}_{\mathsf{DDH}} + 2\Pr[F] = \mathsf{negl}$
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
- The Generic Group Model

Fujisaki-Okamoto

Secure Integration of Asymmetric and Symmetric Encryption Schemes, CRYPTO 1999, JoC 2013

- γ**-spread** and **OWCPA-secure** PKC $(\mathsf{Gen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$
- **one-time secure** cipher (e.g. one-time pad)
- **random oracles** G and H

→ construct a PKC which is INDCCA-secure (many variants possible)

$$\mathsf{Gen} \rightarrow (\mathsf{pk} = \mathsf{pk}_0, \mathsf{sk} = (\mathsf{sk}_0, \mathsf{pk}_0))$$

$$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt}; \mathsf{coins}) \rightarrow \left(\mathsf{Enc}_{0,\mathsf{pk}_0}(\mathsf{coins}; \underbrace{H(\mathsf{coins}, \mathsf{ct}_2)}_{\text{new coins}}), \overbrace{\mathsf{pt} \oplus G(\mathsf{coins})}^{\mathsf{ct}_2} \right)$$

$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1, \mathsf{ct}_2)$:

```
1: $\mathsf{Dec}_{0,\mathsf{sk}_0}(\mathsf{ct}_1) \rightarrow \mathsf{coins}$
2: **if** $\mathsf{ct}_1 \neq \mathsf{Enc}_{0,\mathsf{pk}_0}(\mathsf{coins}; H(\mathsf{coins}, \mathsf{ct}_2))$ **then return** $\perp$
3: **return** $\mathsf{ct}_2 \oplus G(\mathsf{coins})$
```

Security Notions

- **γ-spread**:

$$\forall \mathsf{pk}, \mathsf{pt}, \mathsf{ct} \qquad \Pr[\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt}) = \mathsf{ct}] \leq 2^{-\gamma}$$

- **OWCPA-secure**:
  secure against decryption under chosen plaintext attacks

Proof Sketch

$$\mathsf{OWCPA} \xrightarrow{\mathsf{FO}} \mathsf{INDCCA}$$

(need PKC_0 to be γ-spread as well)

```
$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt})$:
  pick $\sigma$
  $\mathsf{ct}_2 \leftarrow \mathsf{pt} \oplus G(\sigma)$
  $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{0,\mathsf{pk}}(\sigma; H(\sigma, \mathsf{ct}_2))$
  return $(\mathsf{ct}_1, \mathsf{ct}_2)$
```

```
$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1, \mathsf{ct}_2)$:
  $\sigma \leftarrow \mathsf{Dec}_{0,\mathsf{sk}}(\mathsf{ct}_1)$
  if $\sigma = \perp$ then return $\perp$
  if $\mathsf{ct}_1 \neq \mathsf{Enc}_{0,\mathsf{pk}}(\sigma; H(\sigma, \mathsf{ct}_2))$
    then return $\perp$
  $\mathsf{pt} \leftarrow \mathsf{ct}_2 \oplus G(\sigma)$
  return pt
```

- modify the decryption oracle so that it does not use sk but only the oracle tables:
  if there is no $(\sigma, \mathsf{ct}_2, h) \in H$ such that $\mathsf{ct}_1 = \mathsf{Enc}_{0,\mathsf{pk}}(\sigma; h)$, then return $\perp$, otherwise, decrypt by using G
- modify F and G on the challenge σ point

Transform Adapted to Quantum Random Oracles...

Targhi–Unruh, Quantum Security of the Fujisaki-Okamoto and OAEP Transforms, TCC 2016

$$\text{OWCPA} \xrightarrow{\text{TU}} \text{INDCCA}$$

(need PKC_0 to be γ-spread as well)

```
$\text{Enc}_{\text{pk}}(\text{pt})$:
  pick $\sigma$
  $\text{ct}_2 \leftarrow \text{pt} \oplus G(\sigma)$
  $\text{ct}_1 \leftarrow \text{Enc}_{0,\text{pk}}(\sigma; H(\sigma, \text{ct}_2))$
  $\text{ct}_3 \leftarrow H'(\sigma)$
  return $(\text{ct}_1, \text{ct}_2, \text{ct}_3)$
```

```
$\text{Dec}_{\text{sk}}(\text{ct}_1, \text{ct}_2, \text{ct}_3)$:
  $\sigma \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct}_1)$
  if $\sigma = \perp$ then return $\perp$
  if $\text{ct}_1 \neq \text{Enc}_{0,\text{pk}}(\sigma; H(\sigma, \text{ct}_2))$
    then return $\perp$
  if $\text{ct}_3 \neq H'(\sigma)$ then return $\perp$
  $\text{pt} \leftarrow \text{ct}_2 \oplus G(\sigma)$
  return 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$(\mathsf{ct}, \mathsf{pt})$:

```
return $1_{\mathsf{pt}=\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})}$
```

Cvo(ct):

```
return $1_{\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})\neq\perp}$
```

$$\begin{tikzcd}
\text{INDCPA (PKC)} \arrow[dr, "T"] & & \\
& \text{OWPCVA (PKC)} \arrow[r, "U"] & \text{INDCCA (KEM)} \\
\text{OWCPA (PKC)} \arrow[uu, "S^\ell"] \arrow[ur, dotted, "\text{(non tight)}", "T"'] & &
\end{tikzcd}$$

S^ℓ :

```
$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt})$:
  pick $x_1, \ldots, x_\ell$
  $\mathsf{ct}_0 \leftarrow \mathsf{pt} \oplus F(x_1, \ldots, x_\ell)$
  $\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}_{0,\mathsf{pk}}(x_i)$, $i = 1, \ldots, \ell$
  return $(\mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$
$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$:
  $x_i \leftarrow \mathsf{Dec}_{0,\mathsf{sk}}(\mathsf{ct}_i)$, $i = 1, \ldots, \ell$
  $\mathsf{pt} \leftarrow \mathsf{ct}_0 \oplus F(x_1, \ldots, x_\ell)$
  return pt
```

T :

```
$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt})$:
  $\mathsf{ct} \leftarrow \mathsf{Enc}_{0,\mathsf{pk}}(\mathsf{pt}; G(\mathsf{pt}))$
  return ct
$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$:
  $\mathsf{pt} \leftarrow \mathsf{Dec}_{0,\mathsf{sk}}(\mathsf{ct})$
  if $\mathsf{pt} = \perp$ then return $\perp$
  if $\mathsf{ct} \neq \mathsf{Enc}_{0,\mathsf{pk}}(\mathsf{pt}; G(\mathsf{pt}))$
    then return $\perp$
  return pt
```

U :

```
$\mathsf{Enc}_{\mathsf{pk}}$:
  pick pt at random
  $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}_{0,\mathsf{pk}}(\mathsf{pt})$
  $K \leftarrow H(\mathsf{pt}, \mathsf{ct})$
  return $(K, \mathsf{ct})$
$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$:
  $\mathsf{pt} \leftarrow \mathsf{Dec}_{0,\mathsf{sk}}(\mathsf{ct})$
  if $\mathsf{pt} = \perp$ then return $\perp$
  $K \leftarrow H(\mathsf{pt}, \mathsf{ct})$
  return $K$
```

S Transform

$$\mathsf{OWCPA} \xrightarrow{S^\ell} \mathsf{INDCPA}$$

```
Enc_pk(pt):
  1: pick $x_1, \ldots, x_\ell$
  2: $\mathsf{ct}_0 \leftarrow \mathsf{pt} \oplus F(x_1, \ldots, x_\ell)$
  3: $\mathsf{ct}_i \overset{\$}{\leftarrow} \mathsf{Enc}_{0,\mathsf{pk}}(x_i)$, $i = 1, \ldots, \ell$
  4: return $(\mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$
```

```
Dec_sk(ct_0, ..., ct_ℓ):
  1: $x_i \leftarrow \mathsf{Dec}_{0,\mathsf{sk}}(\mathsf{ct}_i)$, $i = 1, \ldots, \ell$
  2: $\mathsf{pt} \leftarrow \mathsf{ct}_0 \oplus F(x_1, \ldots, x_\ell)$
  3: return pt
```

The OWCPA → 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 ℓ to make it smaller (but make encryption more costly).

S Transform (Proof Sketch)

$$\text{OWCPA} \xrightarrow{S^\ell} \text{INDCPA}$$

```
Enc_pk(pt):
  1: pick $x_1, \ldots, x_\ell$
  2: $\mathsf{ct}_0 \leftarrow \mathsf{pt} \oplus F(x_1, \ldots, x_\ell)$
  3: $\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}_{0,\mathsf{pk}}(x_i)$, $i = 1, \ldots, \ell$
  4: return $(\mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$
```

```
Dec_sk(ct_0, ..., ct_ℓ):
  1: $x_i \leftarrow \mathsf{Dec}_{0,\mathsf{sk}}(\mathsf{ct}_i)$, $i = 1, \ldots, \ell$
  2: $\mathsf{pt} \leftarrow \mathsf{ct}_0 \oplus F(x_1, \ldots, x_\ell)$
  3: return pt
```

- add a failure event that the adversary queries $F(x_1, \ldots, x_\ell)$
- construct an OWCPA game in which the challenge ct is put at a random place and completed
- define p_i as the pobability that one F query out of the q ones (taken at random) finds the right x_i given that $(x_1, \ldots, x_{i-1})$ are found and the failure event occurs
- use $\frac{1}{\ell} \sum p_i \geq \left(\prod p_i\right)^{\frac{1}{\ell}}$ and $\prod p_i = \frac{1}{q}$

T Transform (Proof Sketch)

$$\text{INDCPA} \xrightarrow{T} \text{OWPCVA}$$

(need PKC_0 to be γ-spread as well)

```
Enc_pk(pt):
    $\text{ct} \leftarrow \text{Enc}_{0,\text{pk}}(\text{pt}; G(\text{pt}))$
    return ct
```

```
Dec_sk(ct):
    $\text{pt} \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct})$
    if $\text{pt} = \perp$ then return $\perp$
    if $\text{ct} \neq \text{Enc}_{0,\text{pk}}(\text{pt}; G(\text{pt}))$ then return $\perp$
    return pt
```

- remove the Pco oracles which can be simulated (use correctness)
- get rid of Cvo oracles queries which can be simulated from the G table
- other queries answer 0, but with probability $2^{-\gamma}$

U Transform (Proof Sketch)

$$\mathrm{OWPCVA} \xrightarrow{U} \mathrm{INDCCA}_{\mathrm{(KEM)}}$$

```
$\mathrm{Enc}_{\mathrm{pk}}$:
  pick pt at random
  $\mathrm{ct} \xleftarrow{\$} \mathrm{Enc}_{0,\mathrm{pk}}(\mathrm{pt})$
  $K \leftarrow H(\mathrm{pt}, \mathrm{ct})$
  return $(K, \mathrm{ct})$
```

```
$\mathrm{Dec}_{\mathrm{sk}}(\mathrm{ct})$:
  $\mathrm{pt} \leftarrow \mathrm{Dec}_{0,\mathrm{sk}}(\mathrm{ct})$
  if $\mathrm{pt} = \perp$ then return $\perp$
  $K \leftarrow H(\mathrm{pt}, \mathrm{ct})$
  return $K$
```

- define a failure event that the right $(\mathrm{pt}, \mathrm{ct})$ is queried to H
- simulate the decryption query by

```
Dec(ct):
  if $\mathrm{Cvo}(\mathrm{ct}) = 0$ then return $\perp$
  for all $(\mathrm{pt}, \mathrm{ct}, K) \in H$ do
      if $\mathrm{Pco}(\mathrm{pt}, \mathrm{ct}) = 1$ then return $K$
  end for
  return a random $K$ and get ready to update $H$
          (in $H$, check if pt would decrypt...)
```

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 Generic Group Model

Generic Group Algorithms

- Baby-step giant step ▸ slide 193
  make group operations, store in tables, and expect a matching
- Pollard Rho algorithm ▸ slide 194
  make group operations and expect a matching
- why don't we assume that only group operations can be done?

we can prove the hardness of DL, CDH, DDH in generic groups

Shoup Generic Model

- in a group G of order q
- set up a random bijection $\varphi : G \rightarrow \mathbf{Z}_q$ (privately)
- "represent" a group element $x \in G$ by $\varphi(x)$
- show only q and representation of group elements
- give access to an oracle

  ```
  OAdd($u, v$):
    **return** $\varphi(\varphi^{-1}(u) + \varphi^{-1}(y))$
  ```
- can implement scalar multiplication (double-and-add)
- can implement inversion (multiplication by $q - 1$)
- can compare group elements (compare representations)
- can pick a random group element (pick its representation)
- can apply random oracles to group elements
- can implement generic algorithms

Results [Shoup 1997]

In a group of order q, in the Shoup generic model...

Theorem (DL)

Let p be the largest prime factor of q. A generic DL algorithm making $m > 0$ oracle calls has advantage $\mathcal{O}(m^2/p)$.

Theorem (CDH)

Let p be the largest prime factor of q. A generic CDH algorithm making $m > 0$ oracle calls has advantage $\mathcal{O}(m^2/p)$.

Theorem (DDH)

Let p be the smallest prime factor of q. A generic DDH algorithm making $m > 0$ oracle calls has advantage $\mathcal{O}(m^2/p)$.

Maurer Generic Model

- in a group G of order q
- set up an array R to public group elements
- show only q
- give access to oracles

```
OAdd$(i, j, k)$:
  $R[k] \leftarrow R[i] + R[j]$
  **return**
```

```
OIsZero$(i)$:
  **return** $1_{R[i]=0}$
```

- can implement scalar multiplication (double-and-add)
- can implement inversion (multiplication by $q - 1$)
- can compare group elements (IsZero of the difference)
- tricky to pick a random group element
- tricky to implement Baby-step giant-step (dictionary...)
- tricky to implement Pollard ρ (hash...)

Example: DL is Hard in (Maurer) GGM

Theorem (DL)

In a cyclic group of prime order q, a generic (Maurer model) DL algorithm making m OIsZero *calls has advantage at most* $\frac{m+1}{q}$.

Proof. Induction: trivial for $m = 0$ + show how to transform $\mathcal{A}$ into $\mathcal{B}$ who simulates the first OIsZero (one less call):

- DL setup: $R[1] = g$ and $R[2] = X$ (DL: ▸ slide 141)
- set $v_1 = (1, 0)$, $v_2 = (0, 1)$, and other $v_i = (0, 0)$
- follow $\mathcal{A}$ and extend OAdd by $v_k \leftarrow (v_i + v_j) \bmod q$
- simulate the first OIsZero by returning $1_{v_i=0}$
- failure case: $v_i = (\alpha, \beta) \neq 0$ but $\alpha + x\beta = 0 \pmod{q}$
- $\mathsf{Adv}_{\mathcal{A}} \leq \mathsf{Adv}_{\mathcal{B}} + \frac{1}{q}$ by difference lemma ▸ slide 110
- from induction: $\mathsf{Adv}_{\mathcal{B}} \leq \frac{m}{q}$

□

Better Comparison Oracle

```
ONew($i$):
if $R[i]$ is not in List then
    add $R[i]$ at the end of List
end if
return position of $R[i]$ in List
```

- emulatable with OIsZero (m calls become $\frac{m(m-1)}{2}$ calls)
- can implement baby-step giant-step
- can implement Pollard ρ

Theorem (DL)

In a cyclic group of prime order q, a generic (Maurer model) DL algorithm making m ONew calls has advantage at most $\frac{1+m^2/2}{q}$.

Final Step towards Equivalence

- make sure no R value is overwritten
  (use a fresh memory location for every OAdd)
- run ONew after every OAdd
  (keep indices of pairwise different values)
- define φ values by lazy sampling
  (for every new index)

Shoup model		Maurer model
m calls to OAdd	$\longrightarrow$	$\frac{m(m-1)}{2}$ calls to OIsZero

! inputs representing nothing known (random generation)
define φ^{-1} values by lazy sampling
(multiply a generator by a random value until it is new)
$\rightarrow$ blow up in number of queries

Algebraic Group Model

- in a group G of order q
- let $A_1, \ldots, A_n$ be all group elements provided to $\mathcal{A}$ as input
- require $\mathcal{A}$ to release any output X as a vector $(x_1, \ldots, x_n) \in \mathbf{Z}_q^n$ such that $X = x_1 A_1 + \cdots + x_n A_n$
- can do group scalar multiplication, inversion, comparison
- can pick a random group element (pick a representation)
- no problem with random oracles
- can implement generic algorithms
- can simulate a Shoup generic algorithm
- can simulate a Maurer generic algorithm

Results [Fuchsbauer-Kiltz-Loss 2018]

Theorem (DL=CDH in AGM)

For a group of prime order q, in the algebraic group model, DL and CDH are equivalent.

CDH in AGM: $\mathcal{A}(G, xG, yG) \to (u, v, w)$ wins iff
$uG + v(xG) + w(yG) = xyG$, i.e. iff $u + vx + wy = xy$
Proof. Given a DL instance (G, Z), pick $\alpha, \beta \in \mathbf{Z}_q$ and set $X = \alpha Z$, $Y = \beta Z$, run $\mathcal{A}(G, X, Y) \to (u, v, w)$, then solve $u + (\alpha v + \beta w)z = \alpha\beta z^2$ in $z \in \mathbf{Z}_q$. □

Conclusion

- a battery of formal security definitions
- the random oracle model: a tool to idealize hash functions
- the generic group model: a tool to idealize groups

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
- blind signatures: final exam 2022–23 ex3
- IND-CCA ElGamal: final exam 2018–19 ex2
- twin DH problem: final exam 2018–19 ex3