# Public Key Cryptography

Federico Pintore [1]

[1] Mathematical Institute,
Oxford University

# Outline

# Outline

# RSA Encryption Scheme

- Designed by Rivest-Shamir-Adleman in 1977.
- One of the most widely used algorithms today, for both signatures and public key encryption.
- Security requires *hardness of integer factorisation*.

# A few bits of Number Theory

- **Euclidean division**: given two integers $a, b$, with $b \neq 0$, there exist unique $q, r \in \mathbb{Z}$ such that $a = bq + r$, with $0 \leq r < |b|$.
- Given a positive integer $N$ and $a \in \mathbb{Z}$, we denote by $a \pmod N$ the reminder of $a$ when divided by $N$.
- **Integers modulo** $N$: given a positive integer $N$, we define $\mathbb{Z}_N$ as the set $\{[i]_N \mid i = 0, \ldots, N-1\}$, where $[i]_N$ is the subset of all the integers having the same reminder of $i$ when divided by $N$.
- We write $i = j \pmod N$ if $[i]_N = [j]_N$.
- Two binary operations can be defined on $\mathbb{Z}_N$:

$$[a]_N + [b]_N := [a+b]_N, \;\; [a]_N[b]_N := [ab]_N.$$

  It is easy to prove that they are *well defined*.

# A few bits of Number Theory

- $(\mathbb{Z}_N, +)$ is an abelian group ($[0]_N$ is the zero element).
- $[a]_N$ is invertible if there exists $[b]_N \in \mathbb{Z}_N$ s.t. $[a]_N[b]_N = [1]_N$.
- Which are the invertible elements in $\mathbb{Z}_N \setminus \{[0]_N\}$?
- We say that an integer $a$ divides another integer $b$ if $b = ac$ for some $c \in \mathbb{Z}$.
- Given two integers, $a$ and $b$, their greatest common divisor gcd$(a, b)$ is the largest integer dividing both $a$ and $b$.
- Given $a, b \in \mathbb{Z}$, there exist integers $X, Y$ such that $aX + bY =$ gcd$(a, b)$. Furthermore, gcd$(a, b)$ is the smallest positive integer that can be expressed in this way.

# A few bits of Number Theory

- **Proposition**: Let $b, N$ integers, with $b \geq 1$ and $N > 1$. Then $[b]_N$ is invertible if and only if $\gcd(b, N) = 1$ (i.e. $b$ and $N$ are relatively prime).

- The set $\mathbb{Z}_N^* = \{[b]_N \in \mathbb{Z}_N \mid \gcd(b, N) = 1\}$ contains all the invertible elements in $\mathbb{Z}_N \setminus \{[0]_N\}$.

- $(\mathbb{Z}_N^*, \cdot)$ is a group.

- Define $\phi(N)$ as the cardinality of $\mathbb{Z}_N^*$ ($\phi : \mathbb{N} \to \mathbb{N}$ is called *the Euler phi function*).

- If $N$ is a prime, then $\phi(N) = N - 1$. If $N = pq$ is a semi-prime (i.e. it is the product of two primes), then $\phi(N) = (p - 1)(q - 1)$.

# A few bits of Number Theory

- **Proposition**: if $\mathbb{G}$ is a finite abelian group of order $m$, then $g^m = 1$ for each $g \in \mathbb{G}$.
- For each $[a]_N \in \mathbb{Z}_N^*$, we have $([a]_N)^{\phi(N)} = [1]_N$.
- Fix a positive integers $N$ and $e$, with gcd$(e, \phi(N)) = 1$. Then the map:

$$f_e([x]_N) = ([x]_N)^e$$

is a permutation of $\mathbb{Z}_N^*$. Indeed, its inverse is the map $f_d$, with $[d]_{\phi(N)}[e]_{\phi(N)} = [1]_{\phi(N)}$, since $ed = \ell\phi(N) + 1$ and $([x]_N)^{\ell\phi(N)} = [1]_N$.

# The factoring problem

Let GenModulus be a PPT algorithm that, on input $n$, outputs $(N, p, q)$, where $N = pq$ and $p, q$ are $n$-bit primes. (More on generation of primes to come.)

- In the experiment $\text{Factor}_{\mathcal{A},\text{GenModulus}}(n)$, the adversary is given the composite number $N$ output by GenModulus on input $n$, and it has to determine the divisors $p, q$.
- Factoring is hard relative to GenModulus if, for all PPT adversaries $\mathcal{A}$, the success probability in the above experiment is negligible in $n$.
- The factoring assumption is the assumption that there exists a GenModulus relative to which factoring is hard.

# The RSA problem

Let GenRSA be a PPT algorithm that, on input $n$, outputs $(N, p, q, e, d)$, where $N = pq$ - $p, q$ are $n$-bit primes - and $[e]_{\varphi(N)}[d]_{\varphi(N)} = [1]_{\varphi(N)}$.

- In the experiment $\mathsf{RSA} - \mathsf{inv}_{\mathcal{A},\mathsf{GenRSA}}(n)$, GenRSA is run on input $n$. The adversary is given $N$ and $e$ together with a uniform element $[y]_N \in \mathbb{Z}_N^*$. It has to determine $[x]_N \in \mathbb{Z}_N^*$ such that $([x]_N)^e = [y]_N$.
- The RSA problem is hard relative to GenModulus if, for all PPT adversaries $\mathcal{A}$, the success probability in the above experiment is negligible in $n$.
- The RSA assumption is the assumption that there exists a GenRSA relative to which the RSA problem is hard.

# Relationship between RSA and Factoring Assumptions

If $N$ is factored, it is possible to compute $\phi(N)$ and hence $[d]_{\phi(N)} = ([e]_{\phi(N)})^{-1}$.

The other direction is still an open problem! The best we can say is:

**Theorem**

*Given as input a composite integer $N$ and integers $e, d$ such that $[e]_{\phi(N)}[d]_{\phi(N)} = [1]_{\phi(N)}$, there is a PPT algorithm that can output a factor of $N$ except with negligible probability (in $\|N\|$).*

# Plain RSA encryption algorithm

- KeyGen$(n)$: a GenRSA algorithm is run on input $n$. The public key is $(N, e)$, the secret key is $(N, d)$. (Recall that $N = pq$, where $p$ and $q$ are two distinct odd primes, while $[e]_{\varphi(N)}[d]_{\varphi(N)}$ is equal to $[1]_{\varphi(N)}$).
- Enc$((N, e), m \in \mathbb{Z}_N^*)$: it computes the ciphertext $c = m^e$.
- Dec$((N, d), c \in \mathbb{Z}_N^*)$: it computes $m' = c^d$.

**Correctness**: $m' = (m^e)^d = m^{ed} = m^{\ell\varphi(N)+1} = m$.

# Plain RSA security

- The factoring assumption implies that it is computationally infeasible to recover the private key from the public key.
- Solving the factorization problem *might not be necessary* for other goals, such as decrypting without the private key.
- The RSA assumption implies that an eavesdropper cannot recover $m$ from $(N, e, c)$ as long as $m$ is chosen uniformly from $\mathbb{Z}_N^*$.
- “Plain RSA” is insecure!
  - What if $m$ is not chosen uniformly from $\mathbb{Z}_N^*$?
  - What if an attacker learns partial information about $m$?
  - Plain RSA is deterministic, therefore, it is not CPA-secure!

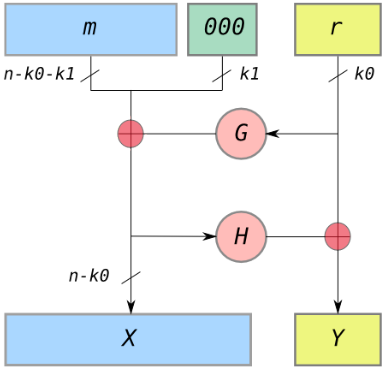
# Padded RSA

- Idea: To encrypt a message $m$, first map it to an element $\tilde{m} \in \mathbb{Z}_n^*$.
- The sender can choose a uniform bit-string $r \in \{0, 1\}^{\ell(n)}$, and sets $\tilde{m} = r||m$ (it is a reversible operation).
- The security of the padded scheme depends on the length $\ell(n)$.
- For instance, $\ell(n) = \mathcal{O}(\log n)$ is a bad choice, since the scheme is not secure in this case.
- The scheme is provably secure based on the RSA problem when $m$ is just a single bit and $\ell$ is very large!
- For other cases, no security proofs based on the RSA problem, BUT no attacks are known either!

# RSA-OAEP

- It is a construction that: is based on the RSA problem, is CCA-secure and uses *optimal asymmetric encryption padding* OAEP.
- Already standardized as a part of RSA PKCS#1 since version 2.0.
- It employs three integer-valued functions $\ell(n), k_0(n), k_1(n)$ with $k_0(n), k_1(n) = \Theta(n)$. There is also a condition on $\ell(n) + k_0(n) + k_1(n)$, which has to be smaller than the minimum bit-length moduli output by GenRSA$(n)$.
- Two hash functions $H$ and $G$ are also used. They are modelled as *random oracles*.
- OAEP is therefore a two-round Feistel network. $G$ and $H$ are the round functions.

# RSA-OAEP




Source: Wikipedia

# RSA-OAEP

Fix $n$ and let $\ell = \ell(n), k_0 = k_0(n), k_1 = k_1(n)$.

Consider $H : \{0,1\}^{\ell+k_1} \to \{0,1\}^{k_0}$ and $G : \{0,1\}^{k_0} \to \{0,1\}^{\ell+k_1}$.

Given a message $m \in \{0,1\}^{\ell}$, the padding is done as follows:

- Set $m' \leftarrow m||0^{k_1}$
- Choose a random $r \in \{0,1\}^{k_0}$
- Compute $s \leftarrow m' \oplus G(r) \in \{0,1\}^{\ell+k_1}$
- Compute $t \leftarrow r \oplus H(s) \in \{0,1\}^{k_0}$
- Finally, set $\tilde{m} \leftarrow s||t$.

# RSA-OAEP

- KeyGen$(n)$: run a GenRSA algorithm on input $n$ to obtain the public key $(N, e)$ and the private key $(N, d)$.

- Enc$((N, e), m)$: pad $m$ to get $\tilde{m}$. The ciphertext will be $c \leftarrow ([\tilde{m}]_N)^e$.

- Dec$((N, d), c)$: compute $\tilde{m} \leftarrow [c]^d$. If $|\tilde{m}| > \ell + k_0 + k_1$, output $\perp$, otherwise;
  - parse $\tilde{m}$ as $s||t$, $s \in \{0, 1\}^{\ell + k_1}, t \in \{0, 1\}^{k_0}$
  - compute $r \leftarrow H(s) \oplus t$
  - compute $m' \leftarrow G(r) \oplus s$. If the least-significant $k_1$ bits of $m'$ are not all 0, output $\perp$. Otherwise, output the $\ell$ **most-significant bits of** $\tilde{m}$.

# Security of RSA-OAEP

- It is CCA-secure assuming that $G$ and $H$ are modelled as random oracles.
- There was an attack on PKCS# v2.0 in 2001 by James Manger, that exploits its implementation - it is a side channel attack!
- The receiver receives the error message $\perp$ in two different cases!
- The time to return the message errors was not identical.
- The attacker can recover a message $m$ using ONLY $|N|$ queries.
- Lesson: side channels attacks are nasty! Implementations should take into consideration every possibility of information leakage!

# RSA weak key generator attack

- Suppose Alice computes a composite number $N_A = pq_A$, while Bob computes $N_B = pq_B$. Is it safe?

# RSA weak key generator attack

- Suppose Alice computes a composite number $N_A = pq_A$, while Bob computes $N_B = pq_B$. Is it safe?
- Everybody sees $N_A := pq_A$ and $N_B := pq_B$.

# RSA weak key generator attack

- Suppose Alice computes a composite number $N_A = pq_A$, while Bob computes $N_B = pq_B$. Is it safe?
- Everybody sees $N_A := pq_A$ and $N_B := pq_B$.
- Alice can compute $q_B = N_B/p$.
- Bob can compute $q_A = N_A/p$.

# RSA weak key generator attack

- Suppose Alice computes a composite number $N_A = pq_A$, while Bob computes $N_B = pq_B$. Is it safe?
- Everybody sees $N_A := pq_A$ and $N_B := pq_B$.
- Alice can compute $q_B = N_B/p$.
- Bob can compute $q_A = N_A/p$.
- **Anyone** can compute $\gcd(N_A, N_B) = p$ and then $q_A$ and $q_A$.
- Attack demonstrated in practice (2012):

  Lenstra et al. *Ron was wrong, Whit is right*

  showed that 2/1000 RSA keys are insecure.

# A CCA secure KEM in the ROM

We consider a KEM consisting of the following algorithms:

- KeyGen$(1^n)$: it runs a GenRSA algorithm on input $n$ to obtain the public key $(N, e)$ and the private key $(N, d)$. It also generates a hash function $H : \mathbb{Z}_N^* \to \{0, 1\}^n$.
- Encaps$(\text{PK}, 1^n)$: it picks a random $r \in \mathbb{Z}_N^*$ and outputs $c \leftarrow r^e$ and the key $k \leftarrow H(r)$.
- Decaps$(\text{SK}, c \in \mathbb{Z}_N^*)$: it first computes $r \leftarrow c^d$ and then outputs $k \leftarrow H(r)$.

This is a part of ISO/IEC18033-2 standard for public-key encryption. Its security relies on the RSA assumption.

# Outline

1. RSA Encryption Scheme
2. Rabin Encryption Scheme
3. Generation of Prime Numbers

# Quadratic Residues

**Definition**

For any positive integer $m$, we define the set of quadratic residues modulo $m$ as

$$QR(m) := \{x \in \mathbb{Z}_m | \quad \exists y \in \mathbb{Z}_m \text{ such that } y^2 = x\}.$$

**Theorem**

*Given a prime $p > 2$, every quadratic residue in $\mathbb{Z}_p^*$ has exactly two square roots (i.e., for each $x \in QR(p) \cap \mathbb{Z}_p^*$ there exist two elements $y, y' \in \mathbb{Z}_p^*$ s.t. $y^2 = (y')^2 = x$.)*

# Quadratic Residues

## Definition

For a prime $p > 2$ and an integer $x$ s.t. $[x]_p \in \mathbb{Z}_p^*$, we define the *Jacobi symbol of $x$ modulo $p$* as follows:

$$\mathcal{J}_p(x) = \begin{cases} +1 & \text{if } [x]_p \in QR(p) \\ -1 & \text{if } [x]_p \notin QR(p). \end{cases}$$

## Theorem

*Given a prime $p > 2$ and an integer $x$ s.t. $[x]_p \in \mathbb{Z}_p^*$, we have* $[\mathcal{J}_p(x)]_p = ([x]_p)^{\frac{p-1}{2}}$.

# Quadratic Residues

**Theorem**

*Let $N = pq$ - where $p$ and $q$ are distinct primes - and let $y$ be an integer such that $[y]_N \in \mathbb{Z}_N^*$. Then $[y]_N$ is a quadratic residue modulo $N$ **iff** $[y]_p$ is a quadratic residue modulo $p$ and $[y]_q$ is a quadratic residue modulo $q$, i.e. $[y]_p \in QR(p)$ and $[y]_q \in QR(q)$.*

**Theorem**

*Let $N = pq$, where $p$ and $q$ are two distinct odd primes. Given $x, \tilde{x}$ s.t. $[x]_N^2 = [y]_N = [\tilde{x}]_N^2$ but $[x]_N \neq \pm[\tilde{x}]_N$, it is possible to factor $N$ in time polynomial in $\|N\|$.*

# Quadratic Residues

**Theorem**

*Let $N = pq$, where $p$ and $q$ are two distinct odd primes such that $[p]_4 = [q]_4 = [3]_4$. Then every quadratic residue modulo $N$ has exactly one square root that belongs to $QR(N)$.*

# Rabin Encryption Scheme

The Rabin encryption scheme consists of the following algorithms:

- KeyGen$(1^n)$: on input $n$, it runs GenModulus$(1^n)$ to obtain $(N, p, q)$ where $N = pq$, $p$ and $q$ are $n$-bit primes with $[p]_4 = [q]_4 = [3]_4$. The public key is $N$, the private key is $(p, q)$.
- Enc(PK, $m \in \{0, 1\}$): it chooses a uniform $[x]_N \in QR(N)$ where $lsb(x) = m$. It outputs the ciphertext $c \leftarrow ([x]_N)^2$.
- Dec(SK, $c$): it computes the unique $[x]_N \in QR(N)$ s.t. $([x]_N)^2 = c$, and outputs $lsb(x)$ (assuming $x < N - 1$).

**Theorem**

*If Factoring is hard relative to* GenModulus*, then this encryption scheme is CPA-secure.*

# Outline

1. RSA Encryption Scheme
2. Rabin Encryption Scheme
3. Generation of Prime Numbers

# Prime numbers

- If a positive integer $a$ divides $b \in \mathbb{Z}$, we call $a$ a divisor of $b$. If $a \notin \{1, b\}$, $a$ is said a non trivial divisor of $b$.
- A positive integer $p$ is prime if it has only trivial divisors.
- There are infinitely many primes.
- **Fundamental Theorem of Arithmetic**: any integer $n$ can be decomposed uniquely has a product of prime numbers.
- **Bertrand's postulate**: for any $n > 1$, the fraction of the $n$-bit integers that are prime is at least $1/3n$.

## Prime numbers

- If a positive integer $a$ divides $b \in \mathbb{Z}$, we call $a$ a divisor of $b$. If $a \notin \{1, b\}$, $a$ is said a non trivial divisor of $b$.
- A positive integer $p$ is prime if it has only trivial divisors.
- There are infinitely many primes.
- **Fundamental Theorem of Arithmetic**: any integer $n$ can be decomposed uniquely has a product of prime numbers.
- **Bertrand's postulate**: for any $n > 1$, the fraction of the $n$-bit integers that are prime is at least $1/3n$.

How to efficiently generate random $n$-bit primes?

# Generating Random Primes

Primes can be generated by picking random $n$-bit integers and checking whether they are prime:

**Algorithm**

*Input: Length* $n$*, parameter* $t$

***For*** $i = 1$ *to* $t$*:*

$p' \leftarrow \{0,1\}^{n-1}$

$p := 1||p'$

***if*** *Primality_test* $(p) = 1$ ***return*** $p$

***return*** *fail*

# Generating Random Primes

- Remember that for any $n > 1$, the fraction of the $n$-bit integers that are prime is at least $1/3n$.
- Now, set $t = 3n^2$. Then the probability that the previous algorithm does not output a prime in $t$ iteration is

$$\left(1 - \frac{1}{3n}\right)^t = \left(\left(1 - \frac{1}{3n}\right)^{3n}\right)^n \leq (e^{-1})^n = e^{-n}$$

- This probability is negligible in $n$.

# Generating Random Primes

- Remember that for any $n > 1$, the fraction of the $n$-bit integers that are prime is at least $1/3n$.
- Now, set $t = 3n^2$. Then the probability that the previous algorithm does not output a prime in $t$ iteration is

$$\left(1 - \frac{1}{3n}\right)^t = \left(\left(1 - \frac{1}{3n}\right)^{3n}\right)^n \leq (e^{-1})^n = e^{-n}$$

- This probability is negligible in $n$.

We still need to study the algorithms that test primality!

# Primality testing

- Given a positive integer $n$, decide whether $n$ is prime or not.
- There are deterministic algorithms for primality testing (see the AKS test, proposed in 2002).
- In practice, we use probabilistic algorithms (having a small probability to return “prime” for composite numbers), since they are much faster.

# Fermat test

- Observation: if $n$ is prime, then $([a]_n)^{n-1} = [1]_n$ for all $[a]_n \in \mathbb{Z}_n^*$ (Fermat's little theorem)
- Idea: choose random $a \in \mathbb{Z}$ and check whether $([a]_n)^{n-1} = [1]_n$. If not, then $n$ is composite.
- We call a *witness that $n$ is composite* any $a \in \mathbb{Z}$ such that $[a]_n \in \mathbb{Z}_n^*$ and $([a]_n)^{n-1} \neq [1]_n$.

# Fermat test

## Algorithm

*Input: Integer $n$, parameter $t$*
***for*** $i = 1$ *to* $t$
  $a \leftarrow \{1, \cdots, n-1\}$
  ***if*** $([a]_n)^{n-1} \neq [1]_n$ ***return*** *"composite"*
***return*** *"prime"*

## Theorem

*If $n$ has a witness that it is composite, then*

$$|\{\textit{witnesses}\}_n| \geq |\mathbb{Z}_n^*|/2.$$

# Fermat test

**Algorithm**

*Input: Integer $n$, parameter $t$*
***for*** $i = 1$ *to* $t$
  $a \leftarrow \{1, \cdots, n-1\}$
  ***if*** $([a]_n)^{n-1} \neq [1]_n$ ***return*** *"composite"*
***return*** *"prime"*

**Theorem**

*If $n$ has a witness that it is composite, then*

$$|\{\textit{witnesses}\}_n| \geq |\mathbb{Z}_n^*|/2.$$

However, try $561$ or $41041$. Observe that the above theorem requires at least a witness!

# Testing Primality

- **Carmichael numbers**: composite numbers that pass the test for all $0 < a < n$, since they don't have any witnesses.

# Testing Primality

- **Carmichael numbers**: composite numbers that pass the test for all $0 < a < n$, since they don't have any witnesses.
- Fermat's test needs to be refined.

# Testing Primality

- **Carmichael numbers**: composite numbers that pass the test for all $0 < a < n$, since they don't have any witnesses.
- Fermat's test needs to be refined.
- Let $n - 1 = 2^k u$, where $u$ is odd and $k \geq 1$ ($n$ is odd).

# Testing Primality

- **Carmichael numbers**: composite numbers that pass the test for all $0 < a < n$, since they don't have any witnesses.
- Fermat's test needs to be refined.
- Let $n - 1 = 2^k u$, where $u$ is odd and $k \geq 1$ ($n$ is odd).
- In Fermat's test, we check if $([a]_n)^{n-1} = ([a]_n)^{2^k u} = [1]_n$.

# Testing Primality

- **Carmichael numbers**: composite numbers that pass the test for all $0 < a < n$, since they don't have any witnesses.
- Fermat's test needs to be refined.
- Let $n - 1 = 2^k u$, where $u$ is odd and $k \geq 1$ ($n$ is odd).
- In Fermat's test, we check if $([a]_n)^{n-1} = ([a]_n)^{2^k u} = [1]_n$.
- What about $([a]_n)^u, ([a]_n)^{2u}, \cdots, ([a]_n)^{2^{k-1}u}$?

# Testing Primality

- **Carmichael numbers**: composite numbers that pass the test for all $0 < a < n$, since they don't have any witnesses.
- Fermat's test needs to be refined.
- Let $n-1 = 2^k u$, where $u$ is odd and $k \geq 1$ ($n$ is odd).
- In Fermat's test, we check if $([a]_n)^{n-1} = ([a]_n)^{2^k u} = [1]_n$.
- What about $([a]_n)^u, ([a]_n)^{2u}, \cdots, ([a]_n)^{2^{k-1}u}$?
- **Strong witness:** $a \in \mathbb{Z}$ is a strong witness that $n$ is composite if $[a]_n \in \mathbb{Z}_n^*$ and
  - $([a]_n)^u \neq \pm[1]_n$
  - $([a]_n)^{2^i u} \neq [-1]_n$ for all $i \in \{1, \cdots, k-1\}$

# Testing Primality

**Theorem**

*Let $n$ be an odd number that is not a prime power. Then we have that at least half of the elements of $\mathbb{Z}_n^*$ are strong witnesses that $n$ is composite.*

Testing whether $n$ is a perfect power (power of an integer, not necessarily prime) can be done in polynomial time!

# Miller-Rabin test

**Algorithm**

*Input: Integer $n > 2$, parameter $t$*
***If*** *$n$ is even,* ***return*** *"composite"*
***If*** *$n$ is a perfect power,* ***return*** *"composite"*
***Write*** *$n - 1 = 2^k u$, where $u$ is odd and $k \geq 1$*
***for*** *$j = 1$ to $t$*
  $a \leftarrow \{1, \cdots, n-1\}$
  ***if*** *$([a]_n)^u \neq \pm[1]_n$ and $([a]_n)^{2^i u} \neq -[1]_n$ for $i \in \{1, \cdots, k-1\}$*
    ***return*** *"composite"*
***return*** *"prime"*

# Miller-Rabin test

**Theorem**

*If $n$ is prime, then the Miller-Rabin test always outputs "prime". If $n$ is composite, the algorithm outputs "composite" except with probability at most $2^{-t}$.*

# Further Reading (1)


- Mihir Bellare, Alexandra Boldyreva, and Silvio Micali.
  Public-key encryption in a multi-user setting: Security proofs and improvements.
  In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer Berlin Heidelberg, 2000.

- Dan Boneh.
  Simplified OAEP for the RSA and Rabin Functions.
  In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer Berlin Heidelberg, 2001.

# Further Reading (2)


- Ronald Cramer and Victor Shoup.
  Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack.
  *SIAM Journal on Computing*, 33(1):167–226, 2003.

- Whitfield Diffie and Martin E Hellman.
  New directions in cryptography.
  *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

# Further Reading (3)


- Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir.
  New attacks on feistel structures with improved memory complexities.
  In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 433–454, 2015.

- Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir.
  Collision-based power analysis of modular exponentiation using chosen-message pairs.
  In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 15–29, 2008.