Trapdoor Permutation Polynomials of $\mathbb{Z}/n\mathbb{Z}$ and Public Key Cryptosystems

Damien Vergnaud

(joint work with Guilhem Castagnos)

Paris, April 26th, 2007

Contents

Public Key Cryptosystems

Alice sends a ciphertext to Bob
Only Bob can recover the plaintext

confidentiality

- To recover the plaintext
  - to find the whole plaintext ?
  - to get some information about it ?
- Which means can be used ?
  - just the public key ?
  - Some extra information ?

Public Key Cryptosystems

Alice sends a ciphertext to Bob
Only Bob can recover the plaintext

confidentiality

- To recover the plaintext
  - to find the whole plaintext ?
  - to get some information about it ?
- Which means can be used ?
  - just the public key ?
  - Some extra information ?

Public Key Cryptosystems

Alice sends a ciphertext to Bob
Only Bob can recover the plaintext

confidentiality

- To recover the plaintext
  - to find the whole plaintext ?
  - to get some information about it ?
- Which means can be used ?
  - just the public key ?
  - Some extra information ?

Why Proving Security?

Once a cryptosystem is described, how can we prove its security?

- by trying to exhibit an attack
  - attack found $\Rightarrow$ system insecure!
  - attack not found $\Rightarrow$?
- by proving that no attack exists under some assumptions
  - attack found $\Rightarrow$ false assumption

If a security proof is given, the system design cannot be incriminated by anyone. But the assumption has to be reasonnable...

Why Proving Security?

Once a cryptosystem is described, how can we prove its security?

- by trying to exhibit an attack
  - attack found $\Rightarrow$ system insecure!
  - attack not found $\Rightarrow$?
- by proving that no attack exists under some assumptions
  - attack found $\Rightarrow$ false assumption

If a security proof is given, the system design cannot be incriminated by anyone. But the assumption has to be reasonnable...

Why Proving Security?

Once a cryptosystem is described, how can we prove its security?

- by trying to exhibit an attack
  - attack found $\Rightarrow$ system insecure!
  - attack not found $\Rightarrow$?
- by proving that no attack exists under some assumptions
  - attack found $\Rightarrow$ false assumption

If a security proof is given, the system design cannot be incriminated by anyone. But the assumption has to be reasonnable...

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally defines a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **model**).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

Examples: OW-PCA, IND-CCA2, NM-CCA2.

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally defines a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **model**).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

Examples: OW-PCA, IND-CCA2, NM-CCA2.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

History of Security Goals

- it shouldn't be feasible to compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitely appeared with public-key crypto.
- it shouldn't be feasible to invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- it shouldn't be feasible to recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- it shouldn't be feasible *to transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

History of Security Goals

- it shouldn't be feasible to compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitely appeared with public-key crypto.
- it shouldn't be feasible to invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- it shouldn't be feasible to recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- it shouldn't be feasible *to transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

History of Security Goals

- it shouldn't be feasible to compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitely appeared with public-key crypto.
- it shouldn't be feasible to invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- it shouldn't be feasible to recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- it shouldn't be feasible *to transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

History of Security Goals

- it shouldn't be feasible to compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitely appeared with public-key crypto.
- it shouldn't be feasible to invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- it shouldn't be feasible to recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- it shouldn't be feasible *to transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario. Rackoff and Simon, 1991.

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

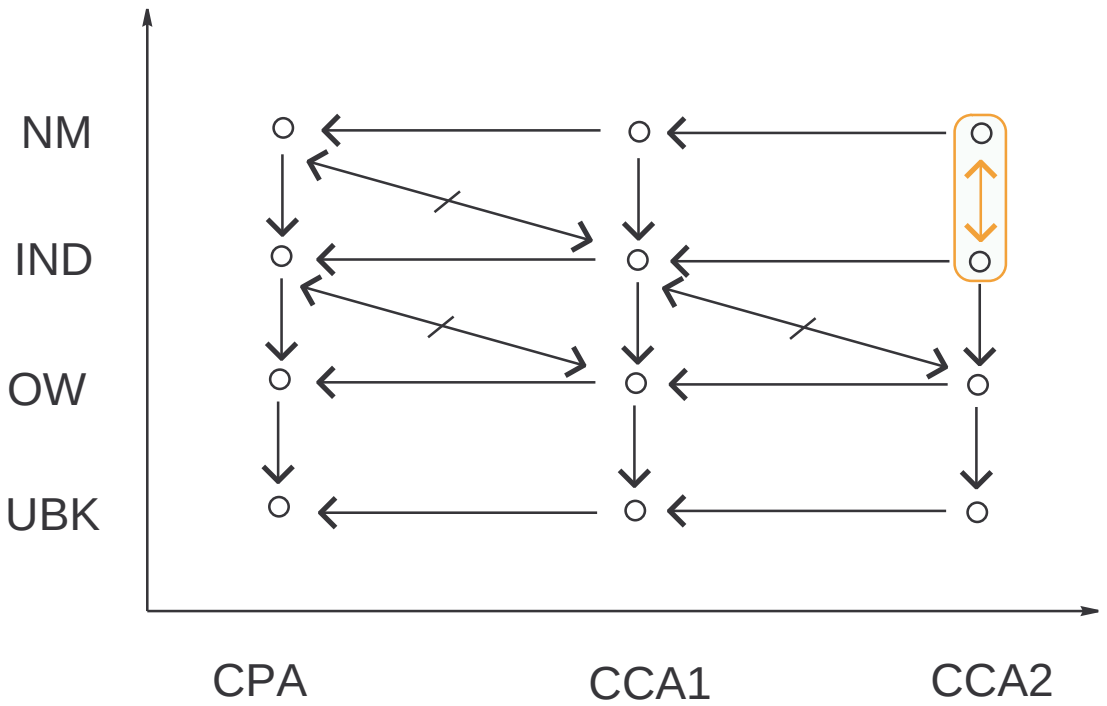
- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario. Rackoff and Simon, 1991.

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario. Rackoff and Simon, 1991.

Relations Among Security Notions



Chosen-Ciphertext Security

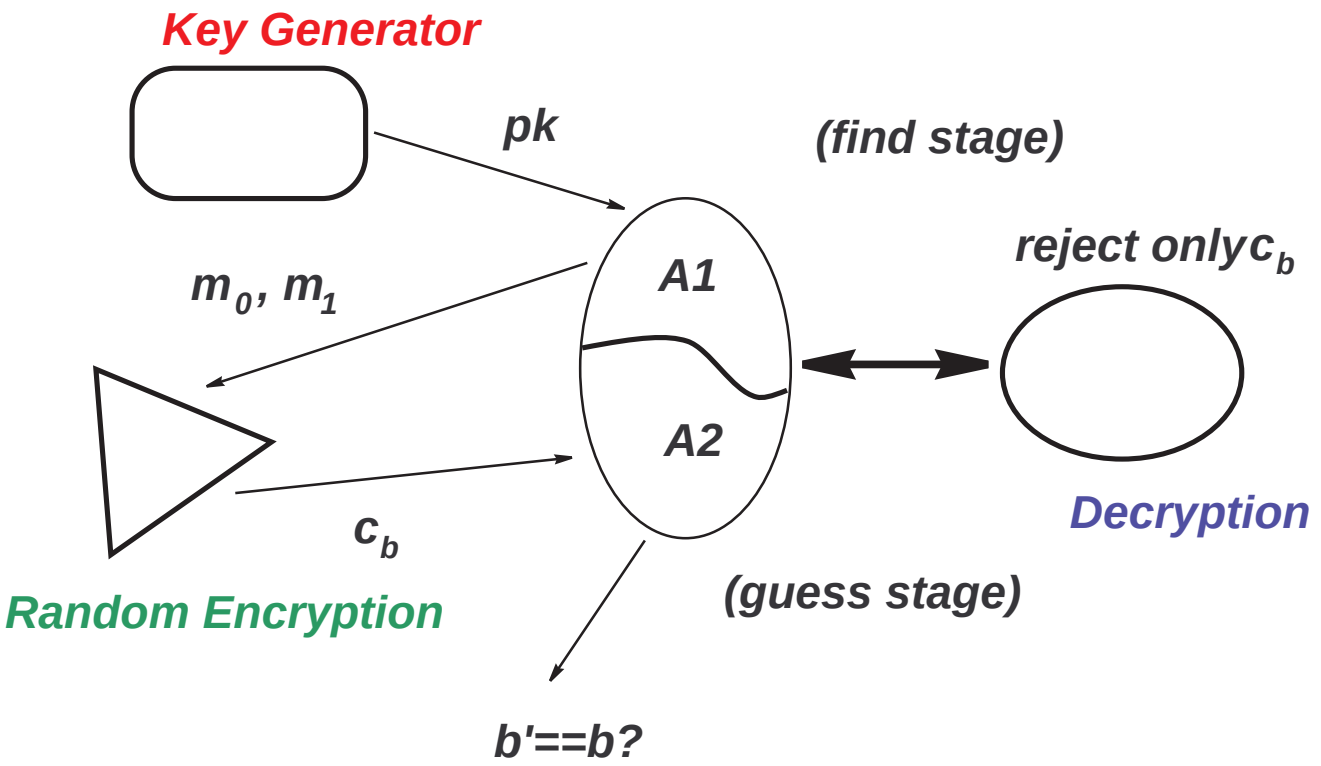
Because IND-CCA2 $\equiv$ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

Formally, an asymmetric encryption scheme is said to be (τ, ε)-IND-CCA if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

$$\mathrm{Adv}^{\mathrm{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \xleftarrow{R} \{0,1\} \\ u \xleftarrow{R} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^{\kappa}), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \ ,$$

where the probability is taken over the random choices of $\mathcal{A}$. The two plaintexts m_0 and m_1 chosen by the adversary have to be of identical length. Access to a decryption oracle is allowed throughout the game.

IND-CCA: Playing the Game



Trapdoor one-way functions

- A **trapdoor permutation** is a one-to-one function f that anyone can compute efficiently; however, inverting f is hard unless some "trapdoor" information is also given.
- Naively, a trapdoor permutation defines a simple public key encryption scheme: the description of f is the public key and the trapdoor is the secret key.
- Unfortunately, the naive public key system is deterministic and hence cannot achieve the *indistinguishability of ciphertexts* security notion.

Trapdoor one-way functions

- A **trapdoor permutation** is a one-to-one function f that anyone can compute efficiently; however, inverting f is hard unless some "trapdoor" information is also given.
- Naively, a trapdoor permutation defines a simple public key encryption scheme: the description of f is the public key and the trapdoor is the secret key.
- Unfortunately, the naive public key system is deterministic and hence cannot achieve the *indistinguishability of ciphertexts* security notion.

Trapdoor one-way functions

- A **trapdoor permutation** is a one-to-one function f that anyone can compute efficiently; however, inverting f is hard unless some "trapdoor" information is also given.
- Naively, a trapdoor permutation defines a simple public key encryption scheme: the description of f is the public key and the trapdoor is the secret key.
- Unfortunately, the naive public key system is deterministic and hence cannot achieve the *indistinguishability of ciphertexts* security notion.

Trapdoor one-way functions: RSA

- In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation.
- The RSA setup consists of choosing two distinct large prime numbers p and q, and computing the RSA modulus $n = pq$.
- The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$).
- The secret key d is defined to be the multiplicative inverse of e modulo $\varphi(n)$.
- Encryption and decryption are defined as follows:

$$\mathcal{E}(m) = m^e \mod n \qquad \mathcal{D}(c) = c^d \mod n.$$

Trapdoor one-way functions: RSA

- In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation.
- The RSA setup consists of choosing two distinct large prime numbers p and q, and computing the RSA modulus $n = pq$.
- The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$).
- The secret key d is defined to be the multiplicative inverse of e modulo $\varphi(n)$.
- Encryption and decryption are defined as follows:

$$\mathcal{E}(m) = m^e \mod n \qquad \mathcal{D}(c) = c^d \mod n.$$

Trapdoor one-way functions: RSA

- In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation.
- The RSA setup consists of choosing two distinct large prime numbers p and q, and computing the RSA modulus $n = pq$.
- The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$).
- The secret key d is defined to be the multiplicative inverse of e modulo $\varphi(n)$.
- Encryption and decryption are defined as follows:

$$\mathcal{E}(m) = m^e \mod n \qquad \mathcal{D}(c) = c^d \mod n.$$

Trapdoor one-way functions: RSA

- In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation.
- The RSA setup consists of choosing two distinct large prime numbers p and q, and computing the RSA modulus $n = pq$.
- The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$).
- The secret key d is defined to be the multiplicative inverse of e modulo $\varphi(n)$.
- Encryption and decryption are defined as follows:

$$\mathcal{E}(m) = m^e \mod n \qquad \mathcal{D}(c) = c^d \mod n.$$

Trapdoor one-way functions: RSA

- In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation.
- The RSA setup consists of choosing two distinct large prime numbers p and q, and computing the RSA modulus $n = pq$.
- The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$).
- The secret key d is defined to be the multiplicative inverse of e modulo $\varphi(n)$.
- Encryption and decryption are defined as follows:

$$\mathcal{E}(m) = m^e \mod n \qquad \mathcal{D}(c) = c^d \mod n.$$

RSA-like functions

- In 1993, Smith and Lennon have proposed a system which uses a special type of Lucas sequences and is an alternative to RSA: given a and b two integers such that $a^2 - 4b$ is a non-square, the Lucas sequence V is given by a second-order linear recurrence relation: $\forall k \geqslant 1$,

$$V_{k+1}(a,b) = aV_k(a,b) - bV_{k-1}(a,b), V_1(a,b) = a, V_0(a,b) = 2.$$

  - The polynomial of degree e, $P(X) \equiv V_e(X,1) \pmod{n}$ with e relatively prime to $(p^2-1)(q^2-1)$ is a permutation of $(\mathbb{Z}/n\mathbb{Z})^\times$ whose inverse is $V_d(X,1) \pmod{n}$ where d is the multiplicative inverse of e modulo $(p^2-1)(q^2-1)$.
- In 1993, Demytko has suggested to replace the polynomials X^e by division polynomials of elliptic curves defined over a ring.

RSA-like functions

- In 1993, Smith and Lennon have proposed a system which uses a special type of Lucas sequences and is an alternative to RSA: given a and b two integers such that $a^2 - 4b$ is a non-square, the Lucas sequence V is given by a second-order linear recurrence relation: $\forall k \geqslant 1$,

$$V_{k+1}(a,b) = aV_k(a,b) - bV_{k-1}(a,b),\ V_1(a,b) = a,\ V_0(a,b) = 2.$$

  - The polynomial of degree e, $P(X) \equiv V_e(X,1) \pmod{n}$ with e relatively prime to $(p^2-1)(q^2-1)$ is a permutation of $(\mathbb{Z}/n\mathbb{Z})^\times$ whose inverse is $V_d(X,1) \pmod{n}$ where d is the multiplicative inverse of e modulo $(p^2-1)(q^2-1)$.

- In 1993, Demytko has suggested to replace the polynomials X^e by division polynomials of elliptic curves defined over a ring.

RSA-like functions

- In 1993, Smith and Lennon have proposed a system which uses a special type of Lucas sequences and is an alternative to RSA: given a and b two integers such that $a^2 - 4b$ is a non-square, the Lucas sequence V is given by a second-order linear recurrence relation: $\forall k \geqslant 1$,

$$V_{k+1}(a,b) = aV_k(a,b) - bV_{k-1}(a,b), V_1(a,b) = a, V_0(a,b) = 2.$$

  - The polynomial of degree e, $P(X) \equiv V_e(X,1) \pmod{n}$ with e relatively prime to $(p^2-1)(q^2-1)$ is a permutation of $(\mathbb{Z}/n\mathbb{Z})^\times$ whose inverse is $V_d(X,1) \pmod{n}$ where d is the multiplicative inverse of e modulo $(p^2-1)(q^2-1)$.
- In 1993, Demytko has suggested to replace the polynomials X^e by division polynomials of elliptic curves defined over a ring.

Trapdoor one-way functions

- These primitives do not provide an IND-CCA2 secure encryption scheme
- Under a slightly stronger assumption than the intractability of the integer factorization, they give a cryptosystem that is only one-way under chosen-plaintext attacks (a very weak level of security).
- The main purpose of this talk is to propose new combinations of these RSA-like problems giving rise to semantically secure public key cryptosystem.
- **Notations.** n will be an RSA modulus and P and Q will denote monic polynomials of $\mathbb{Z}/n\mathbb{Z}[X]$ of respective degree e_P and e_Q, such that the associate polynomial functions are one-way permutations of $(\mathbb{Z}/n\mathbb{Z})^\times$; $R \in \mathbb{Z}/n\mathbb{Z}[X, Y]$ will denote a bivariate polynomial with $e_R := \deg_X(R)$.

Trapdoor one-way functions

- These primitives do not provide an IND-CCA2 secure encryption scheme
- Under a slightly stronger assumption than the intractability of the integer factorization, they give a cryptosystem that is only one-way under chosen-plaintext attacks (a very weak level of security).
- The main purpose of this talk is to propose new combinations of these RSA-like problems giving rise to semantically secure public key cryptosystem.
- **Notations.** n will be an RSA modulus and P and Q will denote monic polynomials of $\mathbb{Z}/n\mathbb{Z}[X]$ of respective degree e_P and e_Q, such that the associate polynomial functions are one-way permutations of $(\mathbb{Z}/n\mathbb{Z})^\times$; $R \in \mathbb{Z}/n\mathbb{Z}[X, Y]$ will denote a bivariate polynomial with $e_R := \deg_X(R)$.

Definitions

- In order to fix the notations, we define the problem of inverting the permutation induced by the polynomial P.
  - **Punctual Inversion:** $P^{-1}(n)$

    Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$;

    Find: $a \in (\mathbb{Z}/n\mathbb{Z})^{\times}$.
- We define a new family of algorithmic problems: the *computation polynomial Diffie-Hellman problems* that generalize together the punctual inversion problem and the dependent-RSA problem.
  - **Computational Polynomial DH:** C-POL-DH(n, P, Q, R)

    Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ and $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$;

    Find: $R(a, b) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$.

Definitions

- In order to fix the notations, we define the problem of inverting the permutation induced by the polynomial P.
  - **Punctual Inversion:** $P^{-1}(n)$

    Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$;

    Find: $a \in (\mathbb{Z}/n\mathbb{Z})^{\times}$.
- We define a new family of algorithmic problems: the *computation polynomial Diffie-Hellman problems* that generalize together the punctual inversion problem and the dependent-RSA problem.
  - **Computational Polynomial DH:** C-POL-DH(n, P, Q, R)

    Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ and $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$;

    Find: $R(a, b) \in (\mathbb{Z}/n\mathbb{Z})^{\times}$.

Definitions

- **Computational Polynomial DH:** C-POL-DH(n, P, Q, R)

  Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$;

  Find: $R(a, b) \in (\mathbb{Z}/n\mathbb{Z})^\times$.

  In this talk, we deal only with the following cases:
  - $R(X, Y) = XY$ that we denote C-POL1(n, P, Q)
  - $R(X, Y) = P((XY)^k)$ that we denote C-POL2(n, k, P, Q)
  - $R(X, Y) = Q(X)$ that we denote C-DPOL(n, P, Q)

Definitions

- **Computational Polynomial DH:** C-POL-DH(n, P, Q, R)

  Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$;

  Find: $R(a, b) \in (\mathbb{Z}/n\mathbb{Z})^\times$.

  In this talk, we deal only with the following cases:
  - $R(X, Y) = XY$ that we denote C-POL1(n, P, Q)
  - $R(X, Y) = P\left((XY)^k\right)$ that we denote C-POL2(n, k, P, Q)
  - $R(X, Y) = Q(X)$ that we denote C-DPOL(n, P, Q)

Definitions

- **Computational Polynomial DH:** C-POL-DH(n, P, Q, R)

  Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$;

  Find: $R(a, b) \in (\mathbb{Z}/n\mathbb{Z})^\times$.

  In this talk, we deal only with the following cases:
  - $R(X, Y) = XY$ that we denote C-POL1(n, P, Q)
  - $R(X, Y) = P\left((XY)^k\right)$ that we denote C-POL2(n, k, P, Q)
  - $R(X, Y) = Q(X)$ that we denote C-DPOL(n, P, Q)

Definitions

- We define the decision problem D-POL-DH(n, P, Q, R) where an element from $(\mathbb{Z}/n\mathbb{Z})^\times$ is given and the algorithm has to decide whether it is a valid candidate for the C-POL-DH(n, P, Q, R) problem.
  - **Decisional Polynomial DH:** C-POL-DH(n, P, Q, R)

    Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$, $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\gamma \in (\mathbb{Z}/n\mathbb{Z})^\times$;

    Decide whether: $\gamma = R(a, b)$.

We also define the decision problems D-POL1(n, P, Q), D-POL2(n, k, P, Q) and D-DPOL(n, P, Q) for the cases $R(X, Y) = XY$, $R(X, Y) = P((XY)^k)$ and $R(X, Y) = Q(X)$.

Definitions

- We define the decision problem D-POL-DH(n, P, Q, R) where an element from $(\mathbb{Z}/n\mathbb{Z})^\times$ is given and the algorithm has to decide whether it is a valid candidate for the C-POL-DH(n, P, Q, R) problem.
  - **Decisional Polynomial DH:** C-POL-DH(n, P, Q, R)

    Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$, $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\gamma \in (\mathbb{Z}/n\mathbb{Z})^\times$;

    Decide whether: $\gamma = R(a, b)$.

  We also define the decision problems D-POL1(n, P, Q), D-POL2(n, k, P, Q) and D-DPOL(n, P, Q) for the cases $R(X, Y) = XY$, $R(X, Y) = P\big((XY)^k\big)$ and $R(X, Y) = Q(X)$.

The C-POL1 problem

We define an extraction problem, E-POL-DH(n, P, Q, R):
Given $P(a)$, $Q(b)$ and $R(a, b)$, find a and b.
We denote as before E-POL1, E-POL2, E-DPOL the extraction problems for the special values of R.

For the C-POL1 problem, we have the straightforward theorem:

Theorem:
D-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longleftarrow}$ C-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1}(n) \wedge Q^{-1}(n)$. □

The C-POL1 problem

We define an extraction problem, E-POL-DH(n, P, Q, R):
Given $P(a)$, $Q(b)$ and $R(a, b)$, find a and b.
We denote as before E-POL1, E-POL2, E-DPOL the extraction problems for the special values of R.

For the C-POL1 problem, we have the straightforward theorem:

Theorem:
D-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longleftarrow}$ C-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1}(n) \wedge Q^{-1}(n)$. □

The C-POL2 problem

For the C-POL2 problem, we use the extraction problem to state a similar theorem.

Theorem:
For an RSA integer n, and two permutation polynomials P and Q of $(\mathbb{Z}/n\mathbb{Z})^\times$,

$$\text{C-POL2} \wedge \text{E-POL2} \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1} \wedge Q^{-1} \overset{\mathcal{P}}{\Longrightarrow} \begin{matrix}\text{C-POL2}\\ \text{E-POL2}\end{matrix} \overset{\mathcal{P}}{\Longrightarrow} \text{D-POL2}\,.$$

□

Difficulty of D-POL1 and D-POL2

- The best known way to solve these problems is to solve the corresponding extraction problem (*cf.* Coppersmith, Franklin, Patarin and Reiter 1996).
- We know the values of $P(a)$, $Q(b)$ and $R(a, b)$ and we want to find the values of a and b. To do this, we compute the resultant with respect to the variable Y:
  $S(X) = \mathrm{Res}_Y(R(X, Y) - R(a, b),\ Q(Y) - Q(b))$.
- This gives a polynomial $S(X)$ of degree $e_R e_Q$ with $S(a) = 0$, so

$$(X - a) \mid \gcd(S(X), P(X) - P(a)).$$

- In fact, in many cases, we will have
  $(X - a) = \gcd(S(X), P(X) - P(a))$, and this method allows to recover a. The value of b is recovered by a symmetric method.

Difficulty of D-POL1 and D-POL2

- The best known way to solve these problems is to solve the corresponding extraction problem (*cf.* Coppersmith, Franklin, Patarin and Reiter 1996).
- We know the values of $P(a)$, $Q(b)$ and $R(a, b)$ and we want to find the values of a and b. To do this, we compute the resultant with respect to the variable Y:
  $S(X) = \mathrm{Res}_Y(R(X, Y) - R(a, b),\ Q(Y) - Q(b))$.
- This gives a polynomial $S(X)$ of degree $e_R e_Q$ with $S(a) = 0$, so

$$(X - a) \mid \gcd(S(X), P(X) - P(a)).$$

- In fact, in many cases, we will have $(X - a) = \gcd(S(X), P(X) - P(a))$, and this method allows to recover a. The value of b is recovered by a symmetric method.

Difficulty of D-POL1 and D-POL2

- The best known way to solve these problems is to solve the corresponding extraction problem (*cf.* Coppersmith, Franklin, Patarin and Reiter 1996).
- We know the values of $P(a)$, $Q(b)$ and $R(a, b)$ and we want to find the values of a and b. To do this, we compute the resultant with respect to the variable Y:
  $S(X) = \mathrm{Res}_Y(R(X, Y) - R(a, b),\ Q(Y) - Q(b))$.
- This gives a polynomial $S(X)$ of degree $e_R e_Q$ with $S(a) = 0$, so

$$(X - a) \mid \gcd(S(X), P(X) - P(a)).$$

- In fact, in many cases, we will have $(X - a) = \gcd(S(X), P(X) - P(a))$, and this method allows to recover a. The value of b is recovered by a symmetric method.

Difficulty of D-POL1 and D-POL2

- The best known way to solve these problems is to solve the corresponding extraction problem (*cf.* Coppersmith, Franklin, Patarin and Reiter 1996).
- We know the values of $P(a)$, $Q(b)$ and $R(a, b)$ and we want to find the values of a and b. To do this, we compute the resultant with respect to the variable Y:
  $S(X) = \mathrm{Res}_Y(R(X, Y) - R(a, b),\ Q(Y) - Q(b))$.
- This gives a polynomial $S(X)$ of degree $e_R e_Q$ with $S(a) = 0$, so

$$(X - a) \mid \gcd(S(X), P(X) - P(a)).$$

- In fact, in many cases, we will have
  $(X - a) = \gcd(S(X), P(X) - P(a))$, and this method allows to recover a. The value of b is recovered by a symmetric method.

Difficulty of D-POL1 and D-POL2

- The computation of the resultant can be done in $\mathcal{O}(e_R^2 e_Q \log^2(e_R e_Q) \log\log(e_R e_Q))$ operations in $\mathbb{Z}/n\mathbb{Z}$.

  Note that $e_R = 1$ for E-POL1 and $e_R = ke_P$ for E-POL2, so, if k is large enough, this method will be infeasible even if e_P is small.
- If e_Q and e_R are greater than, say 2^{21}, this method will fail.

- The computation of the gcd can be done in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R e_Q, e_P)$.
- If e_P and e_Q are greater than, say 2^{60}, this method will fail.

Difficulty of D-POL1 and D-POL2

- The computation of the resultant can be done in $\mathcal{O}(e_R^2 e_Q \log^2(e_R e_Q) \log\log(e_R e_Q))$ operations in $\mathbb{Z}/n\mathbb{Z}$.

  Note that $e_R = 1$ for E-POL1 and $e_R = ke_P$ for E-POL2, so, if k is large enough, this method will be infeasible even if e_P is small.
- If e_Q and e_R are greater than, say 2^{21}, this method will fail.
- The computation of the gcd can be done in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R e_Q, e_P)$.
- If e_p and e_Q are greater than, say 2^{60}, this method will fail.

Difficulty of D-POL1 and D-POL2

- The computation of the resultant can be done in $\mathcal{O}(e_R^2 e_Q \log^2(e_R e_Q) \log\log(e_R e_Q))$ operations in $\mathbb{Z}/n\mathbb{Z}$.

  Note that $e_R = 1$ for E-POL1 and $e_R = ke_P$ for E-POL2, so, if k is large enough, this method will be infeasible even if e_P is small.
- If e_Q and e_R are greater than, say 2^{21}, this method will fail.

- The computation of the gcd can be done in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R e_Q, e_P)$.
- If e_p and e_Q are greater than, say 2^{60}, this method will fail.

Difficulty of D-POL1 and D-POL2

- The computation of the resultant can be done in $\mathcal{O}(e_R^2 e_Q \log^2(e_R e_Q) \log\log(e_R e_Q))$ operations in $\mathbb{Z}/n\mathbb{Z}$.

  Note that $e_R = 1$ for E-POL1 and $e_R = ke_P$ for E-POL2, so, if k is large enough, this method will be infeasible even if e_P is small.
- If e_Q and e_R are greater than, say 2^{21}, this method will fail.

- The computation of the gcd can be done in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R e_Q, e_P)$.
- If e_p and e_Q are greater than, say 2^{60}, this method will fail.

Difficulty of D-POL1 and D-POL2

- The computation of the resultant can be done in $\mathcal{O}(e_R^2 e_Q \log^2(e_R e_Q) \log\log(e_R e_Q))$ operations in $\mathbb{Z}/n\mathbb{Z}$.

  Note that $e_R = 1$ for E-POL1 and $e_R = ke_P$ for E-POL2, so, if k is large enough, this method will be infeasible even if e_P is small.
- If e_Q and e_R are greater than, say 2^{21}, this method will fail.

- The computation of the gcd can be done in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R e_Q, e_P)$.
- If e_p and e_Q are greater than, say 2^{60}, this method will fail.

Difficulty of D-POL1 and D-POL2

If the polynomial Q induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$ (in particular, if Q is associated to the RSA function), it is possible to make another reduction from $Q^{-1}(n)$ to C-POL2(n, k, P, Q) when $k = 1$.

Theorem:
Let n be an RSA integer and P and Q two permutation polynomials of $(\mathbb{Z}/n\mathbb{Z})^\times$ of respective degrees e_P and e_Q. Suppose that Q is a morphism and that $\mathcal{I}$ is the support of P.
If the gcd of $\mathcal{I}$ equals 1, then we can solve the $Q^{-1}(n)$ problem with $\#\mathcal{I}$ queries to an oracle for the C-POL2$(n, 1, P, Q)$ problem with $\mathcal{O}((e_P)^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$.

Difficulty of D-POL1 and D-POL2

Proof:

We denote by m the cardinal of $\mathcal{I}$. Note that if $m = 1$, and $e_Q = e_P$, the problem C-POL2$(n, 1, P, Q)$ is trivial.

Given an element $Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$, we want to recover b.

We start by choosing randomly m couples $(s_j, t_j) \in (\mathbb{Z}/n\mathbb{Z})^\times \times (\mathbb{Z}/n\mathbb{Z})^\times$, with $j = 1, \ldots, m$. We assume that all the s_j and the t_j with $j = 1, \ldots, m$ are distinct.

For each $j \in \{1, \ldots, m\}$, we give the values $P(s_j)$ and $Q(bt_j) = Q(b)Q(t_j)$ to an oracle for C-POL2$(n, 1, P, Q)$ which give the value of $P(s_j t_j b)$ in reply. Note that the m queries to the oracle are independent.

Now we got m equations:

$$\sum_{i \in \mathcal{I}} p_i (s_j t_j)^i b^i = P(s_j t_j b),$$

with the m unknowns $(b^i)_{i \in \mathcal{I}}$.

Difficulty of D-POL1 and D-POL2

Proof:

We denote by m the cardinal of $\mathcal{I}$. Note that if $m = 1$, and $e_Q = e_P$, the problem C-POL2$(n, 1, P, Q)$ is trivial.
Given an element $Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$, we want to recover b.

We start by choosing randomly m couples $(s_j, t_j) \in (\mathbb{Z}/n\mathbb{Z})^\times \times (\mathbb{Z}/n\mathbb{Z})^\times$, with $j = 1, \ldots, m$. We assume that all the s_j and the t_j with $j = 1, \ldots, m$ are distinct.

For each $j \in \{1, \ldots, m\}$, we give the values $P(s_j)$ and $Q(bt_j) = Q(b)Q(t_j)$ to an oracle for C-POL2$(n, 1, P, Q)$ which give the value of $P(s_j t_j b)$ in reply. Note that the m queries to the oracle are independent.

Now we got m equations:

$$\sum_{i \in \mathcal{I}} p_i (s_j t_j)^i b^i = P(s_j t_j b),$$

with the m unknowns $(b^i)_{i \in \mathcal{I}}$.

Difficulty of D-POL1 and D-POL2

Proof:

We denote by m the cardinal of $\mathcal{I}$. Note that if $m = 1$, and $e_Q = e_P$, the problem C-POL2$(n, 1, P, Q)$ is trivial.
Given an element $Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$, we want to recover b.

We start by choosing randomly m couples $(s_j, t_j) \in (\mathbb{Z}/n\mathbb{Z})^\times \times (\mathbb{Z}/n\mathbb{Z})^\times$, with $j = 1, \ldots, m$. We assume that all the s_j and the t_j with $j = 1, \ldots, m$ are distinct.

For each $j \in \{1, \ldots, m\}$, we give the values $P(s_j)$ and $Q(bt_j) = Q(b)Q(t_j)$ to an oracle for C-POL2$(n, 1, P, Q)$ which give the value of $P(s_j t_j b)$ in reply. Note that the m queries to the oracle are independent.

Now we got m equations:

$$\sum_{i \in \mathcal{I}} p_i (s_j t_j)^i b^i = P(s_j t_j b),$$

with the m unknowns $(b^i)_{i \in \mathcal{I}}$.

Difficulty of D-POL1 and D-POL2

Proof:

We denote by m the cardinal of $\mathcal{I}$. Note that if $m = 1$, and $e_Q = e_P$, the problem C-POL2$(n, 1, P, Q)$ is trivial.
Given an element $Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$, we want to recover b.

We start by choosing randomly m couples $(s_j, t_j) \in (\mathbb{Z}/n\mathbb{Z})^\times \times (\mathbb{Z}/n\mathbb{Z})^\times$, with $j = 1, \ldots, m$. We assume that all the s_j and the t_j with $j = 1, \ldots, m$ are distinct.

For each $j \in \{1, \ldots, m\}$, we give the values $P(s_j)$ and $Q(bt_j) = Q(b)Q(t_j)$ to an oracle for C-POL2$(n, 1, P, Q)$ which give the value of $P(s_j t_j b)$ in reply. Note that the m queries to the oracle are independent.

Now we got m equations:

$$\sum_{i \in \mathcal{I}} p_i (s_j t_j)^i b^i = P(s_j t_j b),$$

with the m unknowns $(b^i)_{i \in \mathcal{I}}$.

Difficulty of D-POL1 and D-POL2

Proof (continued):

If we denote $\mathcal{I} := \{i_1, i_2, \ldots, i_m\}$, with $0 < i_1 < i_2 < \cdots < i_m = e_P$, the system of equations is associated with the following matrix:

$$M := \begin{bmatrix} p_{i_1}(s_1 t_1)^{i_1} & p_{i_2}(s_1 t_1)^{i_2} & \cdots & p_{i_m}(s_1 t_1)^{i_m} \\ \vdots & \vdots & & \vdots \\ p_{i_1}(s_m t_m)^{i_1} & p_{i_2}(s_m t_m)^{i_2} & \cdots & p_{i_m}(s_m t_m)^{i_m} \end{bmatrix}$$

The method successes if $\det(M) \in (\mathbb{Z}/n\mathbb{Z})^\times$. We focus on the study of $\det(M) \neq 0$.

If $m = 1$, then $M = ((s_1 t_1)^{e_p})$ so there is no problem, else, we have

$$\det(M) = \left(\prod_{j=1}^{m} p_{i_j}\right) \begin{vmatrix} 1 & c_1^{i_2 - i_1} & \cdots & c_1^{i_m - i_1} \\ \vdots & \vdots & & \vdots \\ 1 & c_m^{i_2 - i_1} & \cdots & c_m^{i_m - i_1} \end{vmatrix}$$

where $c_j := s_j t_j$ for $j = 1, \ldots, m$.

Difficulty of D-POL1 and D-POL2

Proof (continued):

If we denote $\mathcal{I} := \{i_1, i_2, \ldots, i_m\}$, with $0 < i_1 < i_2 < \cdots < i_m = e_P$, the system of equations is associated with the following matrix:

$$M := \begin{bmatrix} p_{i_1}(s_1 t_1)^{i_1} & p_{i_2}(s_1 t_1)^{i_2} & \cdots & p_{i_m}(s_1 t_1)^{i_m} \\ \vdots & \vdots & & \vdots \\ p_{i_1}(s_m t_m)^{i_1} & p_{i_2}(s_m t_m)^{i_2} & \cdots & p_{i_m}(s_m t_m)^{i_m} \end{bmatrix}$$

The method successes if $\det(M) \in (\mathbb{Z}/n\mathbb{Z})^\times$. We focus on the study of $\det(M) \neq 0$.

If $m = 1$, then $M = ((s_1 t_1)^{e_p})$ so there is no problem, else, we have

$$\det(M) = \left(\prod_{j=1}^{m} p_{i_j}\right) \begin{vmatrix} 1 & c_1^{i_2 - i_1} & \cdots & c_1^{i_m - i_1} \\ \vdots & \vdots & & \vdots \\ 1 & c_m^{i_2 - i_1} & \cdots & c_m^{i_m - i_1} \end{vmatrix}$$

where $c_j := s_j t_j$ for $j = 1, \ldots, m$.

Difficulty of D-POL1 and D-POL2

Proof (continued):
This last determinant, D, is a generalized Vandermonde determinant. One can see that

$$D = \left(\prod_{1 \leq i < j \leq m} (c_j - c_i) \right) T(c_1, c_2, \ldots, c_m),$$

where T is a polynomial of degree $i_m - i_1 - m + 1$ in c_m.
So, if all the $(c_j)_{j=1,\ldots,m}$ are distinct, once all the $(s_j)_{j=1,\ldots,m}$, all the $(t_j)_{j=1,\ldots,m-1}$ have been chosen, less than $(i_m - i_1 - m + 1)^2$ values of t_m can make the method fail.
So with standard Gauss elimination, we can recover the $(b^i)_{i \in \mathcal{I}}$ with $\mathcal{O}(e_P^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$ and m independent queries to the oracle.
As $\gcd(\mathcal{I}) = 1$, there exists a linear combination of the elements of $\mathcal{I}$ that equals 1, therefore we can recover b. □

Difficulty of D-POL1 and D-POL2

Proof (continued):
This last determinant, D, is a generalized Vandermonde determinant. One can see that

$$D = \left(\prod_{1 \leq i < j \leq m} (c_j - c_i) \right) T(c_1, c_2, \ldots, c_m),$$

where T is a polynomial of degree $i_m - i_1 - m + 1$ in c_m.
So, if all the $(c_j)_{j=1,\ldots,m}$ are distinct, once all the $(s_j)_{j=1,\ldots,m}$, all the $(t_j)_{j=1,\ldots,m-1}$ have been chosen, less than $(i_m - i_1 - m + 1)^2$ values of t_m can make the method fail.
So with standard Gauss elimination, we can recover the $(b^i)_{i \in \mathcal{I}}$ with $\mathcal{O}(e_P^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$ and m independent queries to the oracle.
As $\gcd(\mathcal{I}) = 1$, there exists a linear combination of the elements of $\mathcal{I}$ that equals 1, therefore we can recover b. □

Difficulty of D-POL1 and D-POL2

Proof (continued):
This last determinant, D, is a generalized Vandermonde determinant.
One can see that

$$D = \left(\prod_{1 \leq i < j \leq m} (c_j - c_i) \right) T(c_1, c_2, \ldots, c_m),$$

where T is a polynomial of degree $i_m - i_1 - m + 1$ in c_m.
So, if all the $(c_j)_{j=1,\ldots,m}$ are distinct, once all the $(s_j)_{j=1,\ldots,m}$, all the $(t_j)_{j=1,\ldots,m-1}$ have been chosen, less than $(i_m - i_1 - m + 1)^2$ values of t_m can make the method fail.
So with standard Gauss elimination, we can recover the $(b^i)_{i \in \mathcal{I}}$ with $\mathcal{O}(e_P^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$ and m independent queries to the oracle.
As $\gcd(\mathcal{I}) = 1$, there exists a linear combination of the elements of $\mathcal{I}$ that equals 1, therefore we can recover b. □

Difficulty of D-POL1 and D-POL2

Proof (continued):
This last determinant, D, is a generalized Vandermonde determinant.
One can see that

$$D = \left(\prod_{1 \leq i < j \leq m} (c_j - c_i) \right) T(c_1, c_2, \ldots, c_m),$$

where T is a polynomial of degree $i_m - i_1 - m + 1$ in c_m.
So, if all the $(c_j)_{j=1,\ldots,m}$ are distinct, once all the $(s_j)_{j=1,\ldots,m}$, all the $(t_j)_{j=1,\ldots,m-1}$ have been chosen, less than $(i_m - i_1 - m + 1)^2$ values of t_m can make the method fail.
So with standard Gauss elimination, we can recover the $(b^i)_{i \in \mathcal{I}}$ with $\mathcal{O}(e_P^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$ and m independent queries to the oracle.
As $\gcd(\mathcal{I}) = 1$, there exists a linear combination of the elements of $\mathcal{I}$ that equals 1, therefore we can recover b. □

The C-DPOL problem

The C-DPOL can be rewritten as follows:
Given $P(a)$, find $Q(a)$;
and the extraction problem, E-DPOL, can be rewritten:
Given $P(a)$ and $Q(a)$, find a.

Theorem: For an RSA integer n, and two permutation polynomials P and Q of $(\mathbb{Z}/n\mathbb{Z})^\times$,

$$\text{C-DPOL} \wedge \text{E-DPOL} \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1} \overset{\mathcal{P}}{\Longrightarrow} \begin{matrix}\text{C-DPOL} \\ \text{E-DPOL}\end{matrix} \overset{\mathcal{P}}{\Longrightarrow} \text{D-DPOL}\,.$$

□

The C-DPOL problem

Let's try to solve the E-DPOL problem. We know the values of $P(a)$ and $Q(a)$ and we want to compute the value of a. We have

$$(X - a) \mid \gcd(P(X) - P(a), Q(X) - Q(a)),$$

and again, in many cases, we will have an equality. The complexity of the computation of the gcd is $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_Q, e_P)$.

Again, suppose that the polynomial P induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$: we can also make another reduction from C-DPOL(n, P, Q) to $P^{-1}(n)$.

Theorem:

Let n be an RSA integer and P and Q two permutation polynomials of $(\mathbb{Z}/n\mathbb{Z})^\times$ of respective degrees e_P and e_Q. Suppose that P is a morphism and that $\mathcal{I}$ is the support of Q.

If $\gcd(\mathcal{I}) = 1$ then we can solve the $P^{-1}(n)$ problem with $\#\mathcal{I}$ queries to an oracle for the C-DPOL(n, P, Q) problem with $\mathcal{O}((e_Q)^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$.

The C-DPOL problem

Let's try to solve the E-DPOL problem. We know the values of $P(a)$ and $Q(a)$ and we want to compute the value of a. We have

$$(X - a) \mid \gcd(P(X) - P(a), Q(X) - Q(a)),$$

and again, in many cases, we will have an equality. The complexity of the computation of the gcd is $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_Q, e_P)$.

Again, suppose that the polynomial P induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$: we can also make another reduction from C-DPOL(n, P, Q) to $P^{-1}(n)$.

Theorem:
Let n be an RSA integer and P and Q two permutation polynomials of $(\mathbb{Z}/n\mathbb{Z})^\times$ of respective degrees e_P and e_Q. Suppose that P is a morphism and that $\mathcal{I}$ is the support of Q.
If $\gcd(\mathcal{I}) = 1$ then we can solve the $P^{-1}(n)$ problem with $\#\mathcal{I}$ queries to an oracle for the C-DPOL(n, P, Q) problem with $\mathcal{O}((e_Q)^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$.

IND-CPA-secure public key cryptosystems

- Let f be a trapdoor permutation and g be another function with the following pseudo-randomness property:

  "The distribution of $(f(k), g(k))$ induced by a random k cannot be distinguished (by a polynomially bounded adversary) from a randomly distributed $(f(k), r)$."

- Then the encryption $E(m) = (f(k), g(k) \oplus m)$ is semantically secure.

- We revisit this approach by using for the function g a trapdoor permutation.

IND-CPA-secure public key cryptosystems

- Let f be a trapdoor permutation and g be another function with the following pseudo-randomness property:

  "The distribution of $(f(k), g(k))$ induced by a random k cannot be distinguished (by a polynomially bounded adversary) from a randomly distributed $(f(k), r)$."

- Then the encryption $E(m) = (f(k), g(k) \oplus m)$ is semantically secure.

- We revisit this approach by using for the function g a trapdoor permutation.

IND-CPA-secure public key cryptosystems

Following this paradigm, we define three new encryption schemes where the public key is (n, P, Q) or (n, P, Q, R) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}). To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^{\times}$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto \big(P(r_0), Q(r_1), mR(r_0, r_1)\big)$

Function 2: $(m, r) \mapsto \big(P(r), mQ(r)\big)$

Function 3: $(m, r) \mapsto \big(P(mr), Q(r^{-1})\big)$

To decrypt, a user uses his knowledge P^{-1} or (P^{-1}, Q^{-1}) to recover r or (r_0, r_1) then m.

IND-CPA-secure public key cryptosystems

Following this paradigm, we define three new encryption schemes where the public key is (n, P, Q) or (n, P, Q, R) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}). To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^{\times}$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto \big(P(r_0), Q(r_1), mR(r_0, r_1)\big)$

Function 2: $(m, r) \mapsto \big(P(r), mQ(r)\big)$

Function 3: $(m, r) \mapsto \big(P(mr), Q(r^{-1})\big)$

To decrypt, a user uses his knowledge P^{-1} or (P^{-1}, Q^{-1}) to recover r or (r_0, r_1) then m.

IND-CPA-secure public key cryptosystems

Following this paradigm, we define three new encryption schemes where the public key is (n, P, Q) or (n, P, Q, R) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}). To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^{\times}$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto \big(P(r_0), Q(r_1), mR(r_0, r_1)\big)$

Function 2: $(m, r) \mapsto \big(P(r), mQ(r)\big)$

Function 3: $(m, r) \mapsto \big(P(mr), Q(r^{-1})\big)$

To decrypt, a user uses his knowledge P^{-1} or (P^{-1}, Q^{-1}) to recover r or (r_0, r_1) then m.

IND-CPA-secure public key cryptosystems

Following this paradigm, we define three new encryption schemes where the public key is (n, P, Q) or (n, P, Q, R) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}). To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^{\times}$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto \big(P(r_0), Q(r_1), mR(r_0, r_1)\big)$

Function 2: $(m, r) \mapsto \big(P(r), mQ(r)\big)$

Function 3: $(m, r) \mapsto \big(P(mr), Q(r^{-1})\big)$

To decrypt, a user uses his knowledge P^{-1} or (P^{-1}, Q^{-1}) to recover r or (r_0, r_1) then m.

IND-CPA-secure public key cryptosystems

The previous schemes are one-way and semantically secure against Chosen Plaintext Attack relative to the following problems:

Encryption function	One-wayness	Semantic security
Function 1, $R(X, Y) = XY$	C-POL1(n, P, Q)	D-POL1(n, P, Q)
Function 1, $R(X, Y) = P\left((XY)^k\right)$	C-POL2(n, k, P, Q)	D-POL2(n, k, P, Q)
Function 2	C-DPOL(n, P, Q)	D-DPOL(n, P, Q)
Function 3	C-POL1(n, P, Q)	D-POL1$(n, P, Q)^{(*)}$

$^{(*)}$ If P or Q is a morphism.

Efficiency considerations

From the encryption functions above, we design five practical cryptosystems, three with Function 1; one with Function 2; and one with Function 3.

For the polynomial P we use the LUC polynomial $V_e(X, 1)$ and for the polynomial Q, the RSA polynomial of the same degree, *i.e.* $Q(X) = X^e$.

In order to compare the efficiency of these schemes, we use an RSA modulus of 1024 bits and we adjust the parameter e (and k) in order to achieve a 2^{80} security (heuristic !!!)

Scheme	Ciphertext	Public keys
Scheme 1	$V_e(r_o, 1), {r_1}^e, mr_0r_1$	$e = 2^{67} + 3.$
Scheme 2	$V_e(r_o, 1), {r_1}^e, mV_e(r_0r_1)$	$e = 2^{23} + 9.$
Scheme 3	$V_e(r_o, 1), {r_1}^e, mV_e\left((r_0r_1)^k\right)$	$e = 5$ and $k = 2^{31} + 65$
Scheme 4	$V_e(r, 1), mr^e$	$e = 2^{67} + 3.$
Scheme 5	$V_e(mr, 1), r^{-e}$	$e = 2^{67} + 3.$

Efficiency considerations

Now, we compare the concrete efficiency of our new schemes.

For the D-RSA scheme of Pointcheval we use $e = 2^{67} + 3$ and for the scheme of Catalano , we use $e = 2^{16} + 1$. The unity of complexity is the cost of a multiplication modulo n.

We use the following estimations: a multiplication modulo n^2 costs as much as three multiplications modulo n, an inversion costs 10 multiplications, a multiplication modulo p costs $1/3$ multiplication modulo n and a multiplication modulo p^2 costs one multiplication modulo n. We use the CRT for the decryption process of all schemes.

Scheme	D-RSA	Catalano	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5
Input	1024						
Output	2048		3072			2048	
Encryption	139	52	205	119	**44**	204	214
Decryption	567	570	1204	1234	1228	736	1196

IND-CCA2-secure public key cryptosystems in the ROM

In the random oracle model, we apply standard techniques to obtain chosen ciphertext security from these new primitives.

The public key is now (n, P, Q, h) or (n, P, Q, R, h) where h is a cryptographic hash function (seen like a random oracle) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}).

To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto (P(r_0), Q(r_1), mR(r_0, r_1), h(m||r_0||r_1))$

Function 2: $(m, r) \mapsto (P(r), mQ(r), h(m||r))$

Function 3: $(m, r) \mapsto (P(mr), Q(r^{-1}), h(m||r))$

The decryption process is done as above except that the message is returned only if the hash value is correct.

IND-CCA2-secure public key cryptosystems in the ROM

In the random oracle model, we apply standard techniques to obtain chosen ciphertext security from these new primitives.

The public key is now (n, P, Q, h) or (n, P, Q, R, h) where h is a cryptographic hash function (seen like a random oracle) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}).

To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto (P(r_0), Q(r_1), mR(r_0, r_1), h(m||r_0||r_1))$

Function 2: $(m, r) \mapsto (P(r), mQ(r), h(m||r))$

Function 3: $(m, r) \mapsto (P(mr), Q(r^{-1}), h(m||r))$

The decryption process is done as above except that the message is returned only if the hash value is correct.

IND-CCA2-secure public key cryptosystems in the ROM

In the random oracle model, we apply standard techniques to obtain chosen ciphertext security from these new primitives.

The public key is now (n, P, Q, h) or (n, P, Q, R, h) where h is a cryptographic hash function (seen like a random oracle) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}).

To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto (P(r_0), Q(r_1), mR(r_0, r_1), h(m||r_0||r_1))$

Function 2: $(m, r) \mapsto (P(r), mQ(r), h(m||r))$

Function 3: $(m, r) \mapsto (P(mr), Q(r^{-1}), h(m||r))$

The decryption process is done as above except that the message is returned only if the hash value is correct.

IND-CCA2-secure public key cryptosystems in the ROM

In the random oracle model, we apply standard techniques to obtain chosen ciphertext security from these new primitives.

The public key is now (n, P, Q, h) or (n, P, Q, R, h) where h is a cryptographic hash function (seen like a random oracle) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}).

To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto (P(r_0), Q(r_1), mR(r_0, r_1), h(m||r_0||r_1))$

Function 2: $(m, r) \mapsto (P(r), mQ(r), h(m||r))$

Function 3: $(m, r) \mapsto (P(mr), Q(r^{-1}), h(m||r))$

The decryption process is done as above except that the message is returned only if the hash value is correct.

IND-CCA2-secure public key cryptosystems in the ROM

In the random oracle model, we apply standard techniques to obtain chosen ciphertext security from these new primitives.

The public key is now (n, P, Q, h) or (n, P, Q, R, h) where h is a cryptographic hash function (seen like a random oracle) and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}).

To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto (P(r_0), Q(r_1), mR(r_0, r_1), h(m||r_0||r_1))$

Function 2: $(m, r) \mapsto (P(r), mQ(r), h(m||r))$

Function 3: $(m, r) \mapsto (P(mr), Q(r^{-1}), h(m||r))$

The decryption process is done as above except that the message is returned only if the hash value is correct.

IND-CCA2-secure public key cryptosystems in the ROM

The previous schemes are semantically secure against Adaptive Chosen Ciphertext Attack in the Random Oracle Model relative to the following problems:

Encryption function	Semantic security
Function 1, $R(X, Y) = XY$	D-POL1(n, P, Q)
Function 1, $R(X, Y) = P\left((XY)^k\right)$	D-POL2(n, k, P, Q)
Function 2	D-DPOL(n, P, Q)
Function 3	D-POL1$(n, P, Q)^{(*)}$

$^{(*)}$ If P or Q is a morphism.

Security proof: Notion of plaintext awareness

- The intuitive idea behind plaintext awareness is that it's hard to construct a new ciphertext for which you can't easily guess the plaintext (or guess that the ciphertext is invalid).
- Such an idea would imply security against chosen ciphertext attack – since the adversary effectively knows the plaintext anyway, the decryption oracle is useless.
- The formalization introduces a *plaintext extractor* – an algorithm which, given a ciphertext and possibly the random oracle queries of the program which created it, returns the corresponding plaintext. □

Conclusion

- We have defined new algorithmic problems, derived from the RSA assumption, and discuss their computational difficulty.
- We have applied them to design public key encryption protocols with IND-CPA-security and IND-CCA2-security in the random oracle model under the assumption of the intractability of their decisional variants.
- The ideas developed in this extended abstract can be used to design encryption schemes with higher security.
  - It is possible to modify our schemes in order to make them IND-CCA2 in the random oracle model relative to the corresponding computational problems.
  - It is possible to construct the most efficient known IND-CCA1-secure encryption scheme with security analysis in the standard security model.
  - In addition, by using the approach proposed by Cramer and Shoup in 2003, we have been able to design a concrete encryption scheme that is proven IND-CCA2-secure in the standard.

Conclusion

- We have defined new algorithmic problems, derived from the RSA assumption, and discuss their computational difficulty.
- We have applied them to design public key encryption protocols with IND-CPA-security and IND-CCA2-security in the random oracle model under the assumption of the intractability of their decisional variants.
- The ideas developed in this extended abstract can be used to design encryption schemes with higher security.
  - It is possible to modify our schemes in order to make them IND-CCA2 in the random oracle model relative to the corresponding computational problems.
  - It is possible to construct the most efficient known IND-CCA1-secure encryption scheme with security analysis in the standard security model.
  - In addition, by using the approach proposed by Cramer and Shoup in 2003, we have been able to design a concrete encryption scheme that is proven IND-CCA2-secure in the standard.

Conclusion

- We have defined new algorithmic problems, derived from the RSA assumption, and discuss their computational difficulty.
- We have applied them to design public key encryption protocols with IND-CPA-security and IND-CCA2-security in the random oracle model under the assumption of the intractability of their decisional variants.
- The ideas developed in this extended abstract can be used to design encryption schemes with higher security.
  - It is possible to modify our schemes in order to make them IND-CCA2 in the random oracle model relative to the corresponding computational problems.
  - It is possible to construct the most efficient known IND-CCA1-secure encryption scheme with security analysis in the standard security model.
  - In addition, by using the approach proposed by Cramer and Shoup in 2003, we have been able to design a concrete encryption scheme that is proven IND-CCA2-secure in the standard.

Conclusion

- We have defined new algorithmic problems, derived from the RSA assumption, and discuss their computational difficulty.
- We have applied them to design public key encryption protocols with IND-CPA-security and IND-CCA2-security in the random oracle model under the assumption of the intractability of their decisional variants.
- The ideas developed in this extended abstract can be used to design encryption schemes with higher security.
  - It is possible to modify our schemes in order to make them IND-CCA2 in the random oracle model relative to the corresponding computational problems.
  - It is possible to construct the most efficient known IND-CCA1-secure encryption scheme with security analysis in the standard security model.
  - In addition, by using the approach proposed by Cramer and Shoup in 2003, we have been able to design a concrete encryption scheme that is proven IND-CCA2-secure in the standard.