Searching for a Secure Public Key Cryptosystem

Simon R. Blackburn

20th September 2012

Outline of this talk

1. Attacks on Textbook RSA
2. Formalising security
3. Some lessons
4. A critique of a group-based PKC

Textbook RSA

Let k be a security parameter.
We generate the public key (integers n, e) and private key (integer d) as follows:

Textbook RSA

Let k be a security parameter.
We generate the public key (integers n, e) and private key (integer d) as follows:

Generate $k/2$-bit primes p, q. Set $n = pq$.

Choose a k-bit integer e coprime to $(p-1)(q-1)$.

Set $d = e^{-1} \bmod (p-1)(q-1)$.

Textbook RSA

Given a message $m \in \mathbb{Z}_n$ and a public key, we encrypt to form a ciphertext c by computing:

$$c = m^e \bmod n.$$

Textbook RSA

Given a message $m \in \mathbb{Z}_n$ and a public key, we encrypt to form a ciphertext c by computing:

$$c = m^e \bmod n.$$

Given a ciphertext $c \in \mathbb{Z}_n$, the public key and the private key, we decrypt to recover the message by computing:

$$m = c^d \bmod n.$$

Textbook RSA

Given a message $m \in \mathbb{Z}_n$ and a public key, we encrypt to form a ciphertext c by computing:

$$c = m^e \bmod n.$$

Given a ciphertext $c \in \mathbb{Z}_n$, the public key and the private key, we decrypt to recover the message by computing:

$$m = c^d \bmod n.$$

This works since

$$(m^e)^d = m \bmod n.$$

Textbook RSA

Given a message $m \in \mathbb{Z}_n$ and a public key, we encrypt to form a ciphertext c by computing:

$$c = m^e \bmod n.$$

Given a ciphertext $c \in \mathbb{Z}_n$, the public key and the private key, we decrypt to recover the message by computing:

$$m = c^d \bmod n.$$

This works since

$$(m^e)^d = m \bmod n.$$

If you can factor n, you can derive d from n and e. The scheme is then broken.

In practice

Choose $k = 1024$, or $k = 2048$ to prevent n being factored.

In practice

Choose $k = 1024$, or $k = 2048$ to prevent n being factored.

For efficiency of encryption, choose e to be small: $e = 2^{16} + 1$ is usual.

In practice

Choose $k = 1024$, or $k = 2048$ to prevent n being factored.

For efficiency of encryption, choose e to be small: $e = 2^{16} + 1$ is usual.

You need a carefully designed randomised message padding scheme.

Textbook RSA is insecure!

Attack 1: If Bob transmits 'Yes' or 'No', Eve can decrypt. (Anyone can encrypt.)

Textbook RSA is insecure!

Attack 1: If Bob transmits 'Yes' or 'No', Eve can decrypt. (Anyone can encrypt.)

Attack 2: The Jacobi symbol of m is not changed by encryption.

Textbook RSA is insecure!

Attack 1: If Bob transmits 'Yes' or 'No', Eve can decrypt. (Anyone can encrypt.)

Attack 2: The Jacobi symbol of m is not changed by encryption.

Attack 3: If m is short, $c = m^e$ (no modular reduction). Decrypt by taking real eth roots.

What is a PKC?

Three public algorithms (probabilistic polynomial time Turing machines):

What is a PKC?

Three public algorithms (probabilistic polynomial time Turing machines):

`KeyGen`

Input: Security parameter (in unary).

Output: Public key, and private key.

What is a PKC?

Three public algorithms (probabilistic polynomial time Turing machines):

`KeyGen`

Input: Security parameter (in unary).

Output: Public key, and private key.

`Enc`

Input: Public key, and message.

Output: Ciphertext.

What is a PKC?

Three public algorithms (probabilistic polynomial time Turing machines):

`KeyGen`

Input: Security parameter (in unary).
Output: Public key, and private key.

`Enc`

Input: Public key, and message.
Output: Ciphertext.

`Dec`

Input: Public key, private key, ciphertext.
Output: Message.

What is a PKC?

Three public algorithms (probabilistic polynomial time Turing machines):

`KeyGen`
Input: Security parameter (in unary).
Output: Public key, and private key.

`Enc`
Input: Public key, and message.
Output: Ciphertext.

`Dec`
Input: Public key, private key, ciphertext.
Output: Message.

For all pairs (pk, sk) output by `KeyGen`, and all messages m,

$$\mathtt{Dec}_{pk,sk}\,\mathtt{Enc}_{pk}(m) = m.$$

What is an adversary?

An adversary `Adv` is (usually) a polynomial time, probabilistic Turing machine.

What is an adversary?

An adversary `Adv` is (usually) a polynomial time, probabilistic Turing machine.

`Adv` always has the public key as an input.

What is an adversary?

An adversary `Adv` is (usually) a polynomial time, probabilistic Turing machine.

`Adv` always has the public key as an input.

`Adv` has access to a decryption oracle. This is the CCA2 model.
(A passive model, with no oracle access, is also studied.)

What is an adversary?

An adversary `Adv` is (usually) a polynomial time, probabilistic Turing machine.

`Adv` always has the public key as an input.

`Adv` has access to a decryption oracle. This is the CCA2 model.
(A passive model, with no oracle access, is also studied.)

Depending on the security model, `Adv` might have other inputs (e.g. a challenge ciphertext).

Defining the Security of a PKC

The minimum security level (for the theoretical cryptographer) is called IND-CCA2.

Defining the Security of a PKC

The minimum security level (for the theoretical cryptographer) is called IND-CCA2. Define a two stage game, with a CCA2 adversary.

Defining the Security of a PKC

The minimum security level (for the theoretical cryptographer) is called IND-CCA2. Define a two stage game, with a CCA2 adversary.

Stage 1: Use `KeyGen` to generate (pk, sk).
`Adv` receives pk as input, and outputs two messages m_0 and m_1.

Defining the Security of a PKC

The minimum security level (for the theoretical cryptographer) is called IND-CCA2. Define a two stage game, with a CCA2 adversary.

Stage 1: Use `KeyGen` to generate (pk, sk).
`Adv` receives pk as input, and outputs two messages m_0 and m_1.

Stage 2: Let $b \in \{0, 1\}$ be random. Set $c = \mathtt{Enc}_{pk}(m_b)$.
`Adv` receives c as input (also a transcript of Stage 1). `Adv` cannot query its decryption oracle on c.
`Adv` outputs a bit b'.

Defining the Security of a PKC

The minimum security level (for the theoretical cryptographer) is called IND-CCA2. Define a two stage game, with a CCA2 adversary.

Stage 1: Use `KeyGen` to generate (pk, sk).
`Adv` receives pk as input, and outputs two messages m_0 and m_1.

Stage 2: Let $b \in \{0, 1\}$ be random. Set $c = \mathtt{Enc}_{pk}(m_b)$.
`Adv` receives c as input (also a transcript of Stage 1). `Adv` cannot query its decryption oracle on c.
`Adv` outputs a bit b'.

`Adv` wins the game if $b = b'$. The PKC is IND-CCA2 secure if for all poly time adversaries and all polynimials p

$$\text{Prob}(\texttt{Adv wins}) < 1/2 + 1/p(k)$$

for sufficiently large k.

Textbook RSA is not IND-CCA2 secure

- `Enc` must be randomised in an IND-CCA2 secure system.

Textbook RSA is not IND-CCA2 secure

- `Enc` must be randomised in an IND-CCA2 secure system.
- The decryption oracle needs to give no useful information to `Adv`. In particular:

Textbook RSA is not IND-CCA2 secure

- `Enc` must be randomised in an IND-CCA2 secure system.
- The decryption oracle needs to give no useful information to `Adv`. In particular:
  - Submitting modified challenge ciphertexts is no good to `Adv`.

Textbook RSA is not IND-CCA2 secure

- `Enc` must be randomised in an IND-CCA2 secure system.
- The decryption oracle needs to give no useful information to `Adv`. In particular:
  - Submitting modified challenge ciphertexts is no good to `Adv`.
  - Submitting random ciphertexts gives no useful information.

RSA can be made secure

Optimal asymmetric encryption padding (OAEP): a randomised padding scheme due to Bellare and Rogaway.

RSA can be made secure

Optimal asymmetric encryption padding (OAEP): a randomised padding scheme due to Bellare and Rogaway.

- Adds randomness;
- Destroys algebraic relationships between messages;
- Checks padded messages are correctly constructed.

RSA can be made secure

Optimal asymmetric encryption padding (OAEP): a randomised padding scheme due to Bellare and Rogaway.

- Adds randomness;
- Destroys algebraic relationships between messages;
- Checks padded messages are correctly constructed.
- RSA-OAEP is IND-CCA2 secure in the random oracle model, provided taking eth roots modulo n is hard.

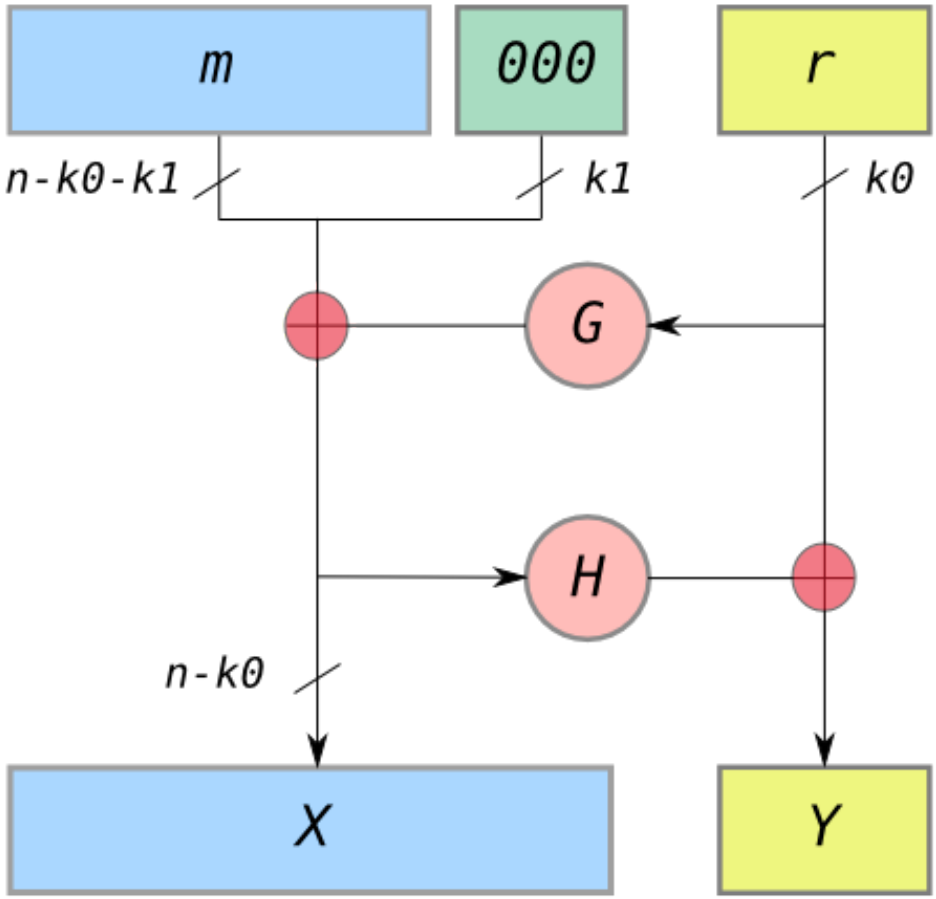
RSA can be made secure

Optimal asymmetric encryption padding (OAEP): a randomised padding scheme due to Bellare and Rogaway.

- Adds randomness;
- Destroys algebraic relationships between messages;
- Checks padded messages are correctly constructed.
- RSA-OAEP is IND-CCA2 secure in the random oracle model, provided taking eth roots modulo n is hard.
- Similar padding schemes exist for discrete log schemes.
- Generic methods (Fujisaki–Okamoto) exist too.

OAEP

... is defined as follows (G, H are hash functions):



Attribution for diagram: Ozga at en.wikipedia

Some Lessons

- Cryptographers are ultimately concerned with creating and understanding practical systems.
- The representation counts: think like a computer.

Some Lessons

- Cryptographers are ultimately concerned with creating and understanding practical systems.
- The representation counts: think like a computer.
- Security is subtle. Cryptographers have thought hard about security for 35 years.
- So it's hard to say something new and interesting about security.

Some Lessons

- Cryptographers are ultimately concerned with creating and understanding practical systems.
- The representation counts: think like a computer.
- Security is subtle. Cryptographers have thought hard about security for 35 years.
- So it's hard to say something new and interesting about security.
- Inventing new primitives (such as one-way functions) that cryptographers know how to build into cryptographic systems will go down well.

What do cryptographers want from mathematicians?

Cryptographers are interested in:

What do cryptographers want from mathematicians?

Cryptographers are interested in:

- Trapdoor one-way permutations. (RSA, for example.)

What do cryptographers want from mathematicians?

Cryptographers are interested in:

- Trapdoor one-way permutations. (RSA, for example.)
- Sets of commuting transformations that are hard to invert, but commute with each other. (Exponentiation in a finite field, for example.)

What do cryptographers want from mathematicians?

Cryptographers are interested in:

- Trapdoor one-way permutations. (RSA, for example.)
- Sets of commuting transformations that are hard to invert, but commute with each other. (Exponentiation in a finite field, for example.)
- Any (computationally) hard problem that becomes easy (for a computer) with some side information.

What do cryptographers want from mathematicians?

Cryptographers are interested in:

- Trapdoor one-way permutations. (RSA, for example.)
- Sets of commuting transformations that are hard to invert, but commute with each other. (Exponentiation in a finite field, for example.)
- Any (computationally) hard problem that becomes easy (for a computer) with some side information.

Good example

Cryptographers love braid groups!

The Osin–Shpilrain scheme

The *Osin–Shpilrain scheme*, introduced to 'explore how non-recursiveness of a decision problem [...] can be used in public key cryptography'.

The Osin–Shpilrain scheme

The *Osin–Shpilrain scheme*, introduced to 'explore how non-recursiveness of a decision problem [...] can be used in public key cryptography'.

Alice publishes presentations for two groups

$$\Gamma_1 = \langle X_1 \mid R_1 \rangle,$$
$$\Gamma_2 = \langle X_2 \mid R_2 \rangle.$$

One Γ_i is trivial, the other is infinite with a word problem that (only) Alice can solve.

The Osin–Shpilrain scheme

The *Osin–Shpilrain scheme*, introduced to 'explore how non-recursiveness of a decision problem [...] can be used in public key cryptography'.

Alice publishes presentations for two groups

$$\Gamma_1 = \langle X_1 \mid R_1 \rangle,$$
$$\Gamma_2 = \langle X_2 \mid R_2 \rangle.$$

One Γ_i is trivial, the other is infinite with a word problem that (only) Alice can solve.

To transmit a bit 0: Bob transmits (w_1, w_2), where w_1 is a random word, and w_2 is a random relation for Γ_2.

The Osin–Shpilrain scheme

The *Osin–Shpilrain scheme*, introduced to 'explore how non-recursiveness of a decision problem [...] can be used in public key cryptography'.

Alice publishes presentations for two groups

$$\Gamma_1 = \langle X_1 \mid R_1 \rangle,$$
$$\Gamma_2 = \langle X_2 \mid R_2 \rangle.$$

One Γ_i is trivial, the other is infinite with a word problem that (only) Alice can solve.

To transmit a bit 0: Bob transmits (w_1, w_2), where w_1 is a random word, and w_2 is a random relation for Γ_2.

To transmit a bit 1: Bob transmits (w_1, w_2), where w_2 is a random word, and w_1 is a random relation for Γ_1.

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group.

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group. Alice assumes an word that is equal to the identity in the group was generated that way.

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group.
Alice assumes an word that is equal to the identity in the group was generated that way.
Alice decrypts with overwhelming probability.

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group.
Alice assumes an word that is equal to the identity in the group was generated that way.
Alice decrypts with overwhelming probability.

Comment

Not a cryptosystem! `KeyGen` not specified.

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group.
Alice assumes an word that is equal to the identity in the group was generated that way.
Alice decrypts with overwhelming probability.

Comment

Not a cryptosystem! `KeyGen` not specified.
Not obvious how to specify `KeyGen`. Also, `Enc` not specified precisely.

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group.
Alice assumes an word that is equal to the identity in the group was generated that way.
Alice decrypts with overwhelming probability.

Comment

Not a cryptosystem! `KeyGen` not specified.
Not obvious how to specify `KeyGen`. Also, `Enc` not specified precisely.

There is a nice idea to make `KeyGen` work:

- choose a random presentation (has solvable word problem by small cancellation);
- add generators to reduce the length of relators (so small cancellation theory no longer applies).

Is it a cryptosystem?

To decrypt: Alice discards the group that is trivial, and determines whether or not the remaining word is the identity in the group.
Alice assumes an word that is equal to the identity in the group was generated that way.
Alice decrypts with overwhelming probability.

Comment

Not a cryptosystem! `KeyGen` not specified.
Not obvious how to specify `KeyGen`. Also, `Enc` not specified precisely.

There is a nice idea to make `KeyGen` work:

- choose a random presentation (has solvable word problem by small cancellation);
- add generators to reduce the length of relators (so small cancellation theory no longer applies).

Let's ignore the issue of making this precise.

What is the security model?

Comment

Not a practical cryptosystem (as the authors admit).

What is the security model?

Comment

Not a practical cryptosystem (as the authors admit).

It is motivated by a non-standard security model:

What is the security model?

Comment

Not a practical cryptosystem (as the authors admit).

It is motivated by a non-standard security model:

We assume the adversary is unbounded.

What is the security model?

Comment

Not a practical cryptosystem (as the authors admit).

It is motivated by a non-standard security model:

We assume the adversary is unbounded.

We assume the adversary does not know `KeyGen`.

What is the security model?

Comment

Not a practical cryptosystem (as the authors admit).

It is motivated by a non-standard security model:

We assume the adversary is unbounded.

We assume the adversary does not know `KeyGen`.

Comment

Dangerous to create an unmotivated security model.

What is the security model?

Comment

Not a practical cryptosystem (as the authors admit).

It is motivated by a non-standard security model:

We assume the adversary is unbounded.

We assume the adversary does not know `KeyGen`.

Comment

Dangerous to create an unmotivated security model.
Shannon's Maxim is violated here.

What is the security model?

Another assumption

Osin and Shpilrain assume that the adversary cannot use the public key to determine which group Γ_i is trivial.

What is the security model?

Another assumption

Osin and Shpilrain assume that the adversary cannot use the public key to determine which group Γ_i is trivial.

- Note that the problem of finding which group is trivial is decidable (given the public key). So this assumption is a significant restriction.

What is the security model?

Another assumption

Osin and Shpilrain assume that the adversary cannot use the public key to determine which group Γ_i is trivial.

- Note that the problem of finding which group is trivial is decidable (given the public key). So this assumption is a significant restriction.
- But how can this be made rigorous? (Is computing an equivalent problem OK?)

What is the security model?

Another assumption

Osin and Shpilrain assume that the adversary cannot use the public key to determine which group Γ_i is trivial.

- Note that the problem of finding which group is trivial is decidable (given the public key). So this assumption is a significant restriction.
- But how can this be made rigorous? (Is computing an equivalent problem OK?)
- We could say that the adversary is not given the public key: but isn't this a symmetric key security model? (Then why not use the one-time pad?)

What is the security model?

Another assumption

Osin and Shpilrain assume that the adversary cannot use the public key to determine which group Γ_i is trivial.

- Note that the problem of finding which group is trivial is decidable (given the public key). So this assumption is a significant restriction.
- But how can this be made rigorous? (Is computing an equivalent problem OK?)
- We could say that the adversary is not given the public key: but isn't this a symmetric key security model? (Then why not use the one-time pad?)
- Maybe the adversary is just given an encryption oracle. A highly non-standard assumption for a public key system!

The perils of not thinking like a computer

Osin and Shpilrain claim that an unbounded adversary cannot decrypt with probability better than 3/4.

The perils of not thinking like a computer

Osin and Shpilrain claim that an unbounded adversary cannot decrypt with probability better than 3/4.

Their argument: With probability 1/2, Bob has sent two identity elements to Alice. Since Eve does not know which group is trivial, Eve gets no information in this case, so can only guess.

The perils of not thinking like a computer

Osin and Shpilrain claim that an unbounded adversary cannot decrypt with probability better than $3/4$.

Their argument: With probability $1/2$, Bob has sent two identity elements to Alice. Since Eve does not know which group is trivial, Eve gets no information in this case, so can only guess.

Why this does not convince: Eve does not want to solve this problem! She is given words (w_1, w_2) and needs to answer the following question:

The perils of not thinking like a computer

Osin and Shpilrain claim that an unbounded adversary cannot decrypt with probability better than $3/4$.

Their argument: With probability $1/2$, Bob has sent two identity elements to Alice. Since Eve does not know which group is trivial, Eve gets no information in this case, so can only guess.

Why this does not convince: Eve does not want to solve this problem! She is given words (w_1, w_2) and needs to answer the following question:

Which w_i is (most likely to have been) a relation of Γ_i generated by Bob?

The perils of not thinking like a computer

Osin and Shpilrain claim that an unbounded adversary cannot decrypt with probability better than 3/4.

Their argument: With probability 1/2, Bob has sent two identity elements to Alice. Since Eve does not know which group is trivial, Eve gets no information in this case, so can only guess.

Why this does not convince: Eve does not want to solve this problem! She is given words (w_1, w_2) and needs to answer the following question:

Which w_i is (most likely to have been) a relation of Γ_i generated by Bob?

This can be solved by simulating Bob's encryption algorithm.

A summary of Osin–Shpilrain

- We have *not 'broken' the Osin–Shpilrain cryptosystem.*

A summary of Osin–Shpilrain

- We have *not 'broken' the Osin–Shpilrain cryptosystem.*
  Not possible until it is reasonably well specified: a (very) common problem with group-based cryptosystems.

A summary of Osin–Shpilrain

- We have *not 'broken' the Osin–Shpilrain cryptosystem.*
  Not possible until it is reasonably well specified: a (very) common problem with group-based cryptosystems.
- *We have criticised the security model.* The model doesn't seem to say much about what a cryptographer means by security.

A summary of Osin–Shpilrain

- We have *not 'broken' the Osin–Shpilrain cryptosystem.*
  Not possible until it is reasonably well specified: a (very) common problem with group-based cryptosystems.
- *We have criticised the security model.* The model doesn't seem to say much about what a cryptographer means by security.
  Non-standard security models are a (fairly) common problem with group-based cryptosystems.

A summary of Osin–Shpilrain

- We have *not 'broken' the Osin–Shpilrain cryptosystem.*
  Not possible until it is reasonably well specified: a (very) common problem with group-based cryptosystems.
- *We have criticised the security model.* The model doesn't seem to say much about what a cryptographer means by security.
  Non-standard security models are a (fairly) common problem with group-based cryptosystems.
- *Motivation is problematic.* Where is the link to an undecidable problem, as promised?

A summary of Osin–Shpilrain

- We have *not 'broken' the Osin–Shpilrain cryptosystem.*
  Not possible until it is reasonably well specified: a (very) common problem with group-based cryptosystems.
- *We have criticised the security model.* The model doesn't seem to say much about what a cryptographer means by security.
  Non-standard security models are a (fairly) common problem with group-based cryptosystems.
- *Motivation is problematic.* Where is the link to an undecidable problem, as promised?
- For the above reasons, mainstream cryptographers are not likely to think this cryptosystem is important.

Some Links

This talk will appear soon on my home page:

`http://www.ma.rhul.ac.uk/sblackburn`

S.R. Blackburn, C. Cid and C. Mullan, 'Group theory in cryptography', Groups St Andrews 2009 in Bath, Volume 1 (CUP, 2011) 133-149.

`http://arxiv.org/abs/0906.5545`

N. Smart, 'Cryptography, an Introduction', 3rd edition.

`http://www.cs.bris.ac.uk/~nigel/`