On the (In)Security of RSA and BLS Signatures

Aris Tentes

(NYU)

Joint Work with: **Yevgeniy Dodis** and **Iftach Haitner**

Digital Signatures

- **Alice** wants to send a message m to **Bob**
- **Bob** wants to make sure that this message was signed by **Alice**

SK PK_{Alice}

- **Alice** computes a signature σ using her secret key SK and sends (m,σ) to **Bob**.
- **Bob,** using **Alice's** public key PK_{Alice}, checks if σ is a valid signature of m.

Security of Digital Signatures

Universal unforgeability

- Consider the following game:
  - **Alice** generates a random pair of keys (SK,PK)
  - **Alice** sends PK and a random message m to **Bob**
  - **Bob** outputs a signature σ.

  Bob wins the game if σ is a valid signature of m.
- A Signature Scheme is Universally Unforgeable if the probability of Bob winning the above game is **negligible**.

Existential Unforgeability against Chosen Message Attacks

- Consider the following game (t-CMA Game):
  - **Alice** generates a random pair of keys (SK,PK)
  - **Alice** sends PK to **Bob**
  - **Bob** is allowed to ask from **Alice** the signatures of t arbitrary messages.
  - **Bob** outputs a pair (m,σ)

  Bob wins the game if σ is a valid signature of m and **Bob** never asked from **Alice** the signature of this m.

Existential Unforgeability against Chosen Message Attacks

- A Signature Scheme is t-Unforgeable Against Chosen Message Attacks if the probability of **Bob** winning the above game is negligible.

- A Signature Scheme is Unforgeable Against Chosen Message Attacks if it is t-Unforgeable for unbounded t.

Simple RSA signatures

- Generation of (SK,PK):
  - Pick two primes p,q and let n=pq
  - Pick a random (prime) number e s.t. gcd(e,ϕ(n))=1
  - Let d be such that ed=1 mod ϕ(n)

SK=d and PK=(e,n)

- Signature of m:

$\sigma = m^d \bmod n$

- Checking validity of σ:

Check if $\sigma^e = m \bmod n$

Security of simple RSA Signatures

- Universal Unforgeability: YES
  - **RSA Assum.:** given (e,n) and a random y it is hard to compute the e-th root of y
    - Producing a valid signature of a random message m is equivalent to computing the e-th root of m.
- Existential Unforgeability: NO
  - Pick a random σ and set $m=\sigma^e$ (remember PK=e)
  - Output (m,σ).

RSA Signatures

- Why don't we 'process' the message m before signing it, to achieve stronger security?

- Let H be a hash function
  - Signing m:

$$\sigma = H(m)^d \bmod n$$

- Security?

Random Oracle Model

- In this model every party is assumed to have **black-box** access to the same oracle.

- This oracle answers every query with a **truly random** response, however identical queries have the same response.

- The oracle can be thought as a **random function**

RSA Signatures

- Why don't we 'process' the message m before signing it, to achieve stronger security?

- Let H be a hash function
  - Signing m:

$$\sigma = H(m)^d \bmod n$$

- **Security**: This Scheme is Existentially Unforgeable against Chosen Message Attacks in the **random oracle model**
  - H is the random oracle

Remarks

- The **Random Oracle Model** is ideal but unrealistic
  - A random function cannot be described efficiently
- In practice hash functions like SHA-1, MD5 are used.
- **RSA Signatures** are used in practice because of their simplicity and their short length.
- Ideally we would like to remove the **Random Oracle** from **RSA Signatures**

Our Results

- [Positive] For every t there exists an efficiently computable Hash Function Family such that the resulting **RSA Signature Scheme** is t-Unforgeable Against Chosen Message Attacks

- [Negative] Using 'standard techniques' we **cannot instantiate** the random oracle with an efficiently computable function (so that **RSA Signature Scheme** remains Existentially Unforgeable against Chosen Message Attacks)

Fully Black-Box Reductions

- A standard technique for proving the security of schemes in cryptography.
- We construct a polynomial time algorithm R which given **black-box** access to any forger F breaks a computational assumption.
  - E.g.: In our (positive) result, for every t
    - Came up with a hash function H
    - Constructed an algorithm R
    - Such that for any F which wins the t-CMA game (of the **RSA Scheme** with H) with non-negligible probability, R^F breaks the RSA assumption.

Black-Box Separation

- Such results rule out the existence of fully black-box reductions.
- What we would like to do here is to show the existence of an oracle B such that
  1. Relative to B, **RSA Signatures** are insecure w.r.t. any hash function H
  2. If the **RSA Assumption** is true, then it remains true even relative to B.
- Why would it suffice?
  - Suppose that there exists a reduction R
  - Then R^B would break the **RSA Assumption**
  - Contradiction because of 2.

Our Separation

- We rule out the existence (of a hash function and) of a specific kind of (fully black-box) reductions.
- These are reductions which exploit the representation of the group elements
  - Remember RSA is defined over a group $\mathbf{Z_n^*}$
- What is the model?

Generic Group Model (GGM)

- Group operations are accessed in a black-box way.
- There is an oracle G
- For every n, G selects a random permutation $\pi: Z_n \rightarrow Z_n$.
- On input (n,a,b), G returns

  $\pi(\pi^{-1}(a)\ \pi^{-1}(b) \bmod n)$
- On input (n,a), G returns $\pi((\pi^{-1}(a))^{-1} \bmod n)$
  - If the operation is not valid it does not return anything.

Remarks

- GGM is widely used in cryptography to prove lower bounds.
    - Hardness of DLP,DHP, etc..
- Most known algorithms and reductions are generic.
- However, not all, like Index Calculus for DLP.

RSA Signatures in GGM

- **RSA Signatures** are defined over GGM in a straightforward way.
- Remember:
  - Signing m:

$$\sigma = H(m)^d \bmod n$$

- In GGM the group operations are done with the help of the GGM oracle G
  - Notice that hash function H might also depend on G

Main Theorem

- Suppose that Factoring is hard. Then there exists **no** hash function H, which makes the **RSA Signature Scheme** Existentially Unforgeable against Chosen Message Attacks in the GGM with a fully **Black-Box** Reduction to the **RSA Assumption**.

About the proof

- We constructed an oracle B,
  - relative to which RSA Signatures in GGM are insecure
  - B is otherwise useless

Oracle B

Useful Definitions (1)

- **Hardwired values:** A value w belongs to the set of *hardwired values* of a computation C, if C^G queries G with an input which contains w, and w has never appeared as an output of G in any previous query in C^G.

Useful Definitions (2)

- **Canonical form:** The canonical form of the output y of an algorithm C^G, is a way to symbolically write y as a product of hardwired values, i.e. $y=\Pi_i w^{a\_i}$.

Example: Suppose that C does the following queries

- $(n,w_1,w_2) = b_1$
- $(n,b_1,w_2) = b_2$

Then the canonical form of b_2 is $w_1 w_2^2$

Description of oracle B

- The input it expects is of the form

$(n,e,H,s_1,\ldots,s_t)$

with $t=|H|$.

1. B checks if $s_i^e=H(i) \bmod n$, namely if s_i is a valid signature of i.
2. B computes the canonical form of each $H(i)$, $H(i)=\Pi_j w_j^{a\_ij}$ and constructs matrix $A=\{a_{ij}\}$.
   1. If $\text{rank}_e A<t$, then it returns $\perp$
   2. Else it returns the factorization of n.

Notice: We can efficiently check if B returns $\perp$ or not.

1. Forging with the help of B is quite easy.

2. We proved that if there exists a circuit R^G which can make B return the factorization of n, then using this R we can either
   - factor n or
   - compress the description of $\pi: Z_n \rightarrow Z_n$ (used by the generic group oracle G)

- To prove the second (and more involved) part we used the Compression Argument introduced in [GT00]

Concluding

- Actually we proved something stronger:
  - There exists no reduction to any assumption (not only the RSA Assumption) which can be expressed in terms of ‘games’.
- Using almost the same techniques we proved similar results for BLS signatures.
- Future work: Generalization of the impossibility result for
  - General exponent e (not only prime)
  - Generic Ring Model

Thank you!