

# How Risky is the Random-Oracle Model?

Gaëtan Leurent and Phong Nguyễn



&



&



*France*

*Aug. 19, 2009*

*Full version on <http://eprint.iacr.org/2008/441>*

# Summary

---

- The Random-Oracle Model (ROM)
- Random-Oracle Instantiations
- Robustness of ROM Signatures with respect to Hash Function Defects



**Random Oracles are Practical:  
A Paradigm for Designing Efficient Protocols**

MIHIR BELLARE\*

PHILLIP ROGAWAY†



ACM CCS '93

# The Random-Oracle Model

# Hash Functions

- Many schemes or protocols use **public hash functions**: not easy to prove strong security properties.
- Usual hash functions:  $\{0,1\}^* \rightarrow \{0,1\}^n$

<i>Hash function</i>	<i>MD5</i>	<i>SHA-1</i>	<i>SHA-2 and SHA-3</i>
<i>n</i>	<i>128</i>	<i>160</i>	<i>224, 256, 384, 512</i>



# What is the ROM?

---

- Goes back to at least [FiatShamir86].
- [BeRo93] popularized the ROM: prove security properties when modeling the hash function as a **random oracle**.
- Popular... but also controversial.

# What is a Random Oracle?

- $H: \{0,1\}^* \rightarrow \{0,1\}^n$
- When  $H(m)$  is requested:
  - Answer **uniformly at random** in  $\{0,1\}^n$ , unless  $m$  has been queried before:  
keep the answers consistent.
- ROM security proofs are able to “**simulate**” a random oracle, where outputs are independent and uniformly distributed. Ex: RSA-FDH.

# RSA-FDH (Full-Domain-Hash)

[BeRo93, BeRo96]

---

- $N = pq$  and  $ed \equiv 1 \pmod{(p-1)(q-1)}$
- $H: \{0,1\}^* \rightarrow \mathbf{Z}/N\mathbf{Z}$  full-domain-hash
- $\text{sign}(m) = H(m)^d \pmod N$

# The ROM Controversy

---

- Many standardized schemes are at best proven secure in the ROM, e.g. **RSA-OAEP** encryption and **RSA-PSS** signature.
- But [CaGoHa98] found **ROM-secure** signature schemes which are **insecure for any (efficient) implementation** of the random oracle. According to [KoMe07], all such ROM counterexamples are “artificial”.

# How is [CaGoHa98] Possible?

---

- Any efficient implementation can **be simulated** by a universal Turing machine. This allows a scheme to **decide** whether or not the hash function is a random oracle.
- Then disclose the secret key if the hash is not a random oracle.



# Contradicting the ROM

- If you have an attack against a ROM-secure scheme:
  - Either you can break the computational assumptions of the security proof.
  - Either you exploit a property of the hash function, which is **not shared** by the random-oracle simulation, like in [CaGoHa98].

# Difference between ROM and SM

---

- In the standard model (SM), a security proof gives you a list of **sufficient assumptions** to guarantee security properties.
- In the ROM, **no precise sufficient assumption** on the hash function is provided, except one which cannot be satisfied by efficient functions. The ROM is a security model, not an assumption.

# This Talk

---

- New Issues on the ROM
  - Instantiating a random-oracle with large outputs: problems in existing proposals.
  - Comparing ROM schemes is tricky: **hash function requirements** and **impact of hash defects** can vary a lot.

# How Risky is the ROM?

---

- [Coron\*08] showed that the ROM is equivalent to the Ideal Cipher Model (ICM).
- The ICM is risky:
  - MD5 was **collision-resistant in the ICM**.
  - Yesterday, AES-256 was shown to “differ” substantially from an ideal cipher.
- How about the ROM? More and more hash functions are shown to “differ” from a random oracle...

# Instantiating a Random Oracle

# The ROM Heuristic

---

- When implementing a ROM-secure scheme, you instantiate the random oracle, and **hope** that the scheme will remain secure.
- If the output length is standard (between 128 and 512), a natural candidate is a standard hash function, even if it is known to have weaknesses.

# The Large-Output Case

---

- But many ROM-secure schemes require a hash function with **large output**  $> 512$  bits. Ex: RSA-FDH.
- How are we supposed to implement such functions in practice? Not with MD5 or SHA: **problem not covered in textbooks**, and **often ignored** in papers.

# Proposals for Large-Output

---

- **Bellare and Rogaway**: one in [BeRo93], and another in [BeRo96] (on RSA-FDH and PSS).
- Implicit instantiations in **PKCS and IEEE P1363 standards**, based on SHA-1.
- “Semi-proposals” by [Coron\*05] based on indifferntiability theory [Maurer\*04].

# Our Results

- All these instantiations fall short of the security of a random oracle.
- For 1024 bits:
  - A practical preimage attack on [BeRo93] costing  $2^{30}$ .
  - A collision attack on [BeRo96] costing  $2^{106}$ .
- When applied to MD5/SHA-1, finding collisions on PKCS/IEEE and [Coron\*05] is not more expensive than for MD5/SHA-1, independently of the output length:  $2^{16}$  compression calls for MD5 and  $2^{61}$  for SHA-1.

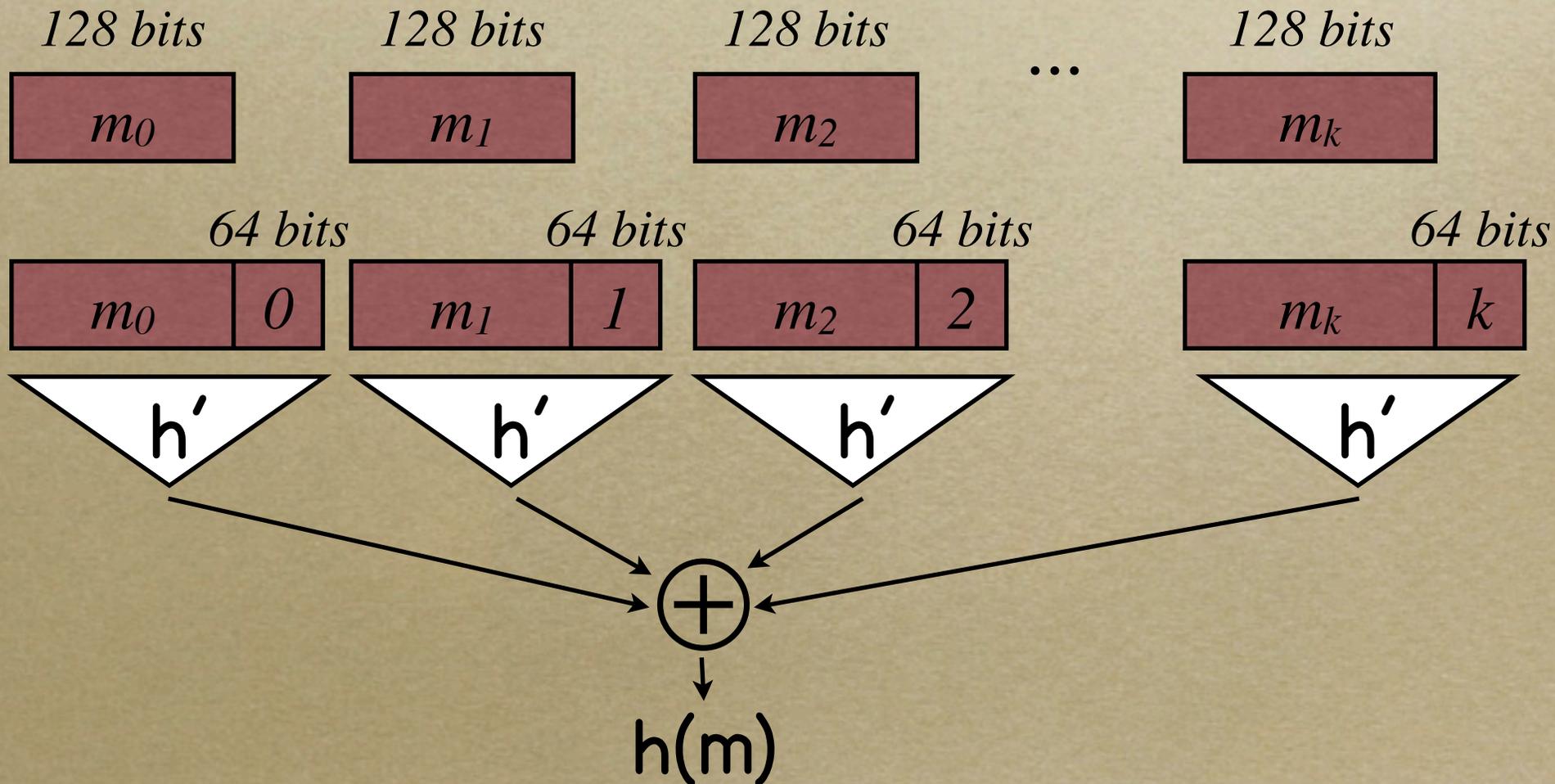
# The Case of [BeRo93]

---

- Complex construction, based on the MD5 compression function.
- Previously, we had a  $2^{67}$  preimage attack for 1024-bit digests, based on **Wagner's generalized birthday** [Wa02].
- Thanks to **[Bellare09]**, we have a  $2^{30}$  preimage attack for 1024-bit digests, using **[BeMi97]** on XHASH. The attack is **polynomial** in the output size.

# Overview of [BeRo93]

- For  $h: \{0,1\}^* \rightarrow \{0,1\}^n$ , build  $h': \{0,1\}^{192} \rightarrow \{0,1\}^n$  using the MD5 compression function.



# A Practical Attack on [BeRo93]

---

- Goal: given  $t \in \{0,1\}^n$ , find a “random”  $m$  s.t.  $h(m) = t$ .
- Finding random preimages only costs  $n^3$ .
  - the preimages have bit-length  $O(n)$ .
  - If  $n=1024$ , the cost is  $2^{30}$ .
- The attack works by linear algebra over  $GF(2)$ .

# A Practical Attack on [BeRo93]

- Goal: given  $t \in \{0,1\}^n$ , find a “random”  $m$  s.t.  $h(m) = t$ .
- Select two random 128-bit  $c[0]$  and  $c[1]$ .
- Now, for any  $x = (x_0, x_1, \dots, x_{n-1}) \in \{0,1\}^n$ ,  
let  $m[x] =$ 

$c[x_0]$	$c[x_1]$	$c[x_2]$	...	$c[x_{n-1}]$
----------	----------	----------	-----	--------------
- Then  $h(m[x]) = t$  can be rewritten as a linear system with  $GF(2)$  unknowns  $x_i$ 's, where the matrix coeffs are the bits of  $h'(c[0] \parallel i) \oplus h'(c[1] \parallel i)$  for each  $i$ .

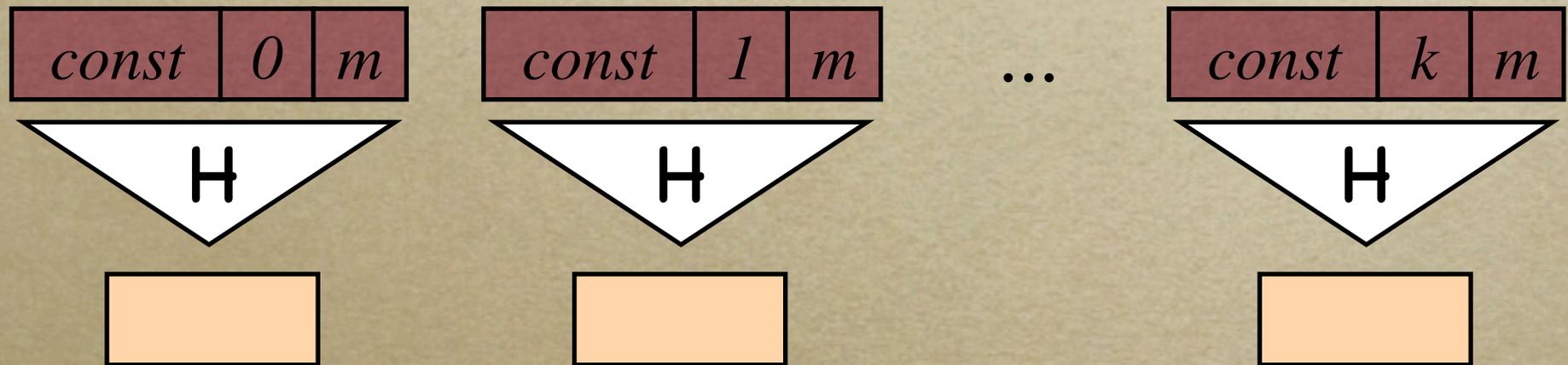
# More on [BeRo93]

---

- In fact, the preimage attack can be generalized to a chosen-prefix (resp. chosen-suffix) preimage attack, with the same cost.
- Goal: given  $t \in \{0,1\}^n$  and  $s \in \{0,1\}^*$ , find  $m$  s.t.  $h(m \parallel s) = t$ .

# Overview of [BeRo96]

- Let  $H=MD5$  or  $SHA-1$ .



- Since *H* is a MD-hash, this is also the concatenation of distinct MD-hashes.

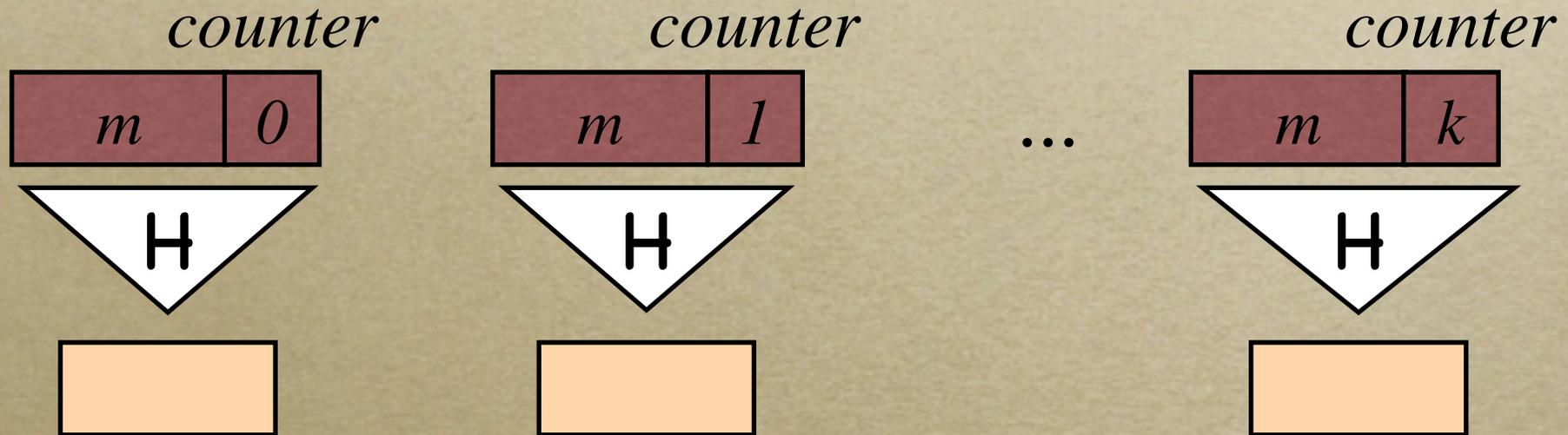
# Attacking [BeRo96]

---

- [Joux04] can attack concatenations of MD-hashes: roughly the same security as a single MD-hash.
- With a tighter analysis of [Joux04], for  $H=MD5$  and 1024-bit output:
  - Collisions in  $2^{106}$ .
  - Preimages in  $2^{166}$ .

# MGF1 in PKCS Standard

- Let  $H=SHA-1$  or  $SHA-2$ .



- Since  $H$  is a MD-hash, any (appropriate-size) collision in  $H$  is a collision for the big hash.

# A Few Words on Indifferentiability

- Following the **indifferentiability framework** [Maurer\*04], many papers [Coron\*05, etc.] give RO-preserving constructions: from a “small” RO, you can obtain a “bigger” RO.



- But no clear proposal for the “small” RO.

# A Few Words on Indifferentiability

- In fact, if you plug MD5/SHA-1 components as the “small” RO in [Coron\*05], the big RO is **as bad as MD5/SHA-1: independently of the output size**, you can find collisions for essentially the same cost.
- Everything depends on the “small” RO.



# Recap

---

- For large output ( $> 512$  bits), there is currently no candidate with the collision-resistance and the preimage-resistance of a random-oracle.
- For instance, [BeRo93] is completely insecure: random preimages and collisions for “free”.

# Robustness of ROM Signatures

# Signature Schemes

---

- One of the first applications of the ROM: “Prove” the security of efficient signature schemes.
- Two main families of ROM signatures:
  - Based on trapdoor OWF: RSA, Rabin, ESIGN, etc.
  - Based on ID-schemes, using the Fiat-Shamir heuristic: Schnorr, etc.

# In this paper/talk

---

- In the paper, we analyze the **hash function requirements** and **impact of hash function defects** for the main ROM signatures based on trapdoor OWF.
- In this talk, we only focus on **derandomization**: for certain schemes, any collision suffices **to disclose the secret key!**

# Derandomization

- Many signature schemes are **probabilistic**: random nonce required for each signature generation.
- **Derandomization** makes them **deterministic**:
  - Proposed by [Granboulan02] to fix the security proof of ESIGN for the NESSIE project.
  - Discussed by [KaWa03] to make schemes stateless.
  - Used by [Bernstein08] and [Boneh\*07] to obtain deterministic ROM-signatures based on factoring, with a **tight security proof**: variants of Rabin and Rabin/Williams.

# How to Derandomize

---

- Generate the random nonce **deterministically** from the message, the secret key, and possibly additional secrets.
- But it has to be done carefully: several methods proposed in [Granboulan02,KaWa03,Boneh\*07].

# Pitfalls in Derandomization

---

- **Soundness**: the ROM security proof must be preserved.
- This is not the case with [Granboulan02]: we give counter-examples where one can find **two messages generating the same nonce**, in which case you can **recover the secret key** with a chosen-message attack on ESIGN or DSA.

# How to Derandomize

---

- [Bernstein08] does not say exactly how it should be performed.
- [KaWa03] discuss several (sound) possibilities, one of which being used in [Boneh\*07]:
  - select the nonce as  $r = F_K(m)$  where  $F$  is a PRF and  $K$  is an additional secret key.

# Our Results

---

- We notice that if ever **one obtains a hash collision**, then one can recover:
  - the master key with a chosen-ID attack on the ID-based cryptosystem of [Boneh\*07].
  - the secret key with a chosen-message on [Bernstein08] using only two messages.

# Explanation

---

- Here, a hash collision does not imply a nonce collision.
- Hence, a hash collision gives rise to **two random square roots of the same public number**, thus disclosing the factorization!

# Surprisingly

---

- The attack can be prevented by slightly modifying the derandomization process, while still preserving the ROM security proof.
- We thus obtain two very close ROM-secure schemes:
  - One of them becomes totally insecure if there is a hash collision.
  - The other does not.

# Resistance to Preimages

---

- But independently of the derandomization, both [Be08] and [BGH07] do not tolerate preimages or malleability.
- So if one plugs [BeRo93] as the random oracle, there are key-recovery attacks on [Be08, BGH07]. Similarly for IEEE P1363's deterministic Rabin-Williams.

# Comparing ROM-secure schemes

---

- Cryptanalysis provides a useful criterion for assessing ROM schemes: evaluating the robustness with respect to RO defects.
- For instance, among all RSA signatures with tight ROM security proofs, RSA-PSS seems the most robust one.

# Random-Oracle Lessons

---

- Proofs in the RO are difficult to compare, even if the hardness assumptions are the same.
  - Tightness in the RO can be misleading.
- More work is needed on how to instantiate a RO.
- The ROM is not an assumption: it is not formalized, and ROM proofs may require totally different properties on the hash function.

# CONCLUSION

# Conclusion

---

- The Random-Oracle Model is useful to **detect design flaws**: if you cannot prove security in the ROM, not a good sign.
- However, it does not provide much “granularity”, which makes comparisons tricky and perhaps, risky.

# Conclusion

---

- Different ROM schemes can have **totally different requirements** on the hash function, with **different impacts**: in one case, a collision can be deadly; in another case, even preimages do not seem to threaten.
- And this is independent of usual ROM criterions: tightness, efficiency.

# Conclusion

---

- Based on MD5/SHA-1, it might be better to select ROM schemes which are the **least risky** with respect to potential hash function defects (such as for large outputs).
- But is it possible to formalize this? For many ROM schemes, a security proof in the standard model is known to be unlikely.

# Open Problems

---

- Here, we focused on signature schemes based on **trapdoor one-way functions**, but can other similar examples be found?
- For instance, public-key encryption: there are many ways to make RSA encryption secure in the ROM, but what happens if the hash functions have defects?

Thank you for your attention...



Any question(s)?