# Flaws in Applying Proof Methodologies to Signature Schemes

**Jacques Stern - David Pointcheval**
Ecole normale supérieure - France

**John Malone-Lee - Nigel Smart**
University of Bristol - UK

## Summary

- The methodology of “provable security”
- The context of signature schemes
  - definitions
  - questions
- Our findings
  - ESIGN
  - ECDSA
- Conclusions

# Provable security: a short story

- Originated in the seminal papers
  [GM86] and [GMR88]
- Received increased applicability by allowing random oracles as a substitute to hash functions
  [FS86, BR93]
- Now requested to support emerging standards
  (IEEE P1363, Cryptrec, NESSIE, ISO)



# The need for provable security

- “Textbook” crypto schemes cannot be used as such
  (obvious homomorphic properties…)
- Practitioners need formatting rules to ensure interoperability
- Heuristic redundancy is not enough
  - attack against PKCS#1 V 1.5 [Bl98]
  - attack against ISO 9796-1 [CNS99, CHJ99]

# The limits of provable security

- Provable security does not yield proofs
  - proofs are relative
  - proofs often use random oracles.
    Meaning is debatable [CGH98]
  - proofs are not formal objects
    but appear in talks and papers.
    Time is needed for acceptance.
- Still, provable security is a means to provide some form of guarantee that a crypto scheme is not flawed



# Provable security in five steps

1 - Define goal of adversary

2 - Define security model

3 - Provide a proof by reduction

4 - Check proof

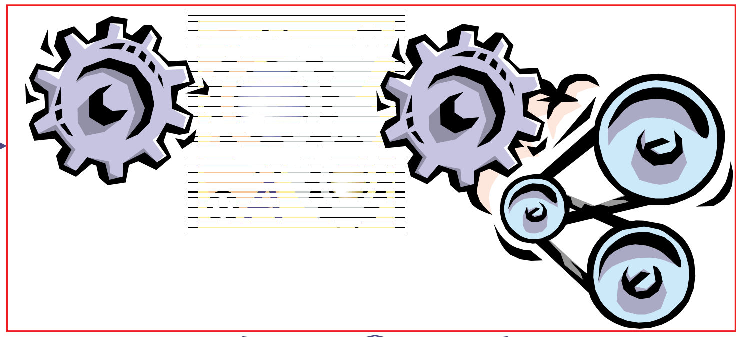
5 - Interpret proof

# Proof by reduction

Reduction of a problem **P** to an attack *Atk*:

- Let *A* be an adversary that breaks the scheme then *A* can be used to solve **P**

Instance **I** of **P** →



→ Solution of **I**

**P** intractable ⇒ scheme unbreakable



# Why other steps matter: OAEP

Proposed formatting standard for RSA encryption [BR94]

1 - Goal of adversary: distinguish random encryptions of two messages $m_0$ $m_1$

2 - Security models: CPA, CCA1, CCA2

3 - Proof (in [BR94])

4 - Does not achieve CCA2 [Sh01]

5 - Alternative proof [FOPS01], specific to RSA-OAEP

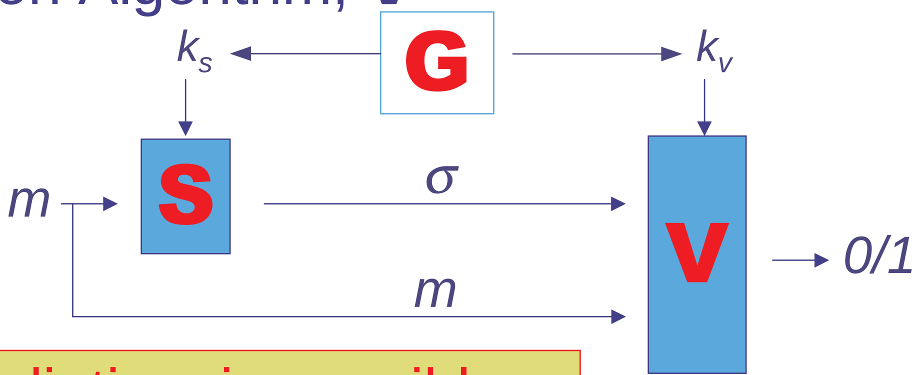
# Signature

- Appends to a message a proof of origin
- This should provide non-repudiation and thus even convince a third party



# Signature scheme

- Key Generation Algorithm **G**
- Signature Algorithm, **S**
- Verification Algorithm, **V**




Non-repudiation: impossible to forge valid $\sigma$ without $k_s$

# Goal of the adversary

- Existential Forgery:

  Try to forge a valid message-signature pair without the private key

  Adversary is successful if the following probability is large

$$\mathrm{Succ}^{ef}(\mathcal{A}) = \Pr\left[\mathbf{V}(m,\sigma) = 1 \middle| \mathcal{A}(k_v) = (m,\sigma)\right]$$



# Security models

- No-Message Attacks: the adversary only knows the verification (public) key
- Known-Message Attacks (KMA): the adversary has access to a list $\Lambda$ of message/signature pairs
- Chosen-Message Attacks (CMA): the messages are adaptively chosen by the adversary

  $\Rightarrow$ the strongest attack

# Q1: submit the same message?

- In a probabilistic signature scheme, several signatures may correspond to a message
- In the usual definition for Existential Forgery in Chosen-Message Attacks (CMA), the adversary can repeatedly submit a message.

Otherwise, weaker model :

- Single-Occurrence Chosen-Message Attacks (SO-CMA) - each message $m$ can be submitted only once; this produces a signature $\sigma$ and $(m, \sigma)$ is added to the list $\Lambda$



# Q2: control key generation?

- In the usual definition for Existential Forgery, it is assumed that key generation **G** is fairly played
- Having the adversary control **G** can affect non-repudiation by allowing duplicate signatures: two different messages $m_1$, $m_2$ with a common $\sigma$
- One can produce $(m_1,\sigma)$ and later claim that $(m_2,\sigma)$ was meant

# Q3: output the same message?

- In the usual definition for Existential Forgery, output forgery corresponds to a fresh message $m$. No pair $(m\ \sigma)$ can be in the list $\Lambda$.

Otherwise, weaker goal:

- Malleability: produce a new pair $(m,\sigma) \notin \Lambda$ possibly for a submitted message
  $((m,\sigma')$ in $\Lambda$ for some $\sigma' \neq \sigma)$
- Non-malleability is a *stronger demand* than resistance to existential forgeries



# ESIGN

A signature scheme designed in the late 90ies and considered in IEEE P1363, Cryptrec NESSIE, together with a security proof

- Uses RSA integers of the form $n=p^2q$
- Based on the Approximate e-*th* root problem: given $y$ find $x$ such that $y \# x^e \bmod n$
- Signature generation is a very efficient way to compute $\sigma = x$, given $y = H(m)$

# Our findings on ESIGN

- Proofs holds only in SO-CMA scenario
- Reduction simulates signature requests by having $x$ ready beforehand such that

  $$H(m) \# x^e \bmod n$$

- Gets stuck if $m$ is queried anew
- Interpretation:
  - ESIGN is not broken
  - either give up CMA property…
  - or modify ESIGN
    (cf. NESSIE internal paper by L. Granboulan)



# ECDSA

**G** $=<\mathbf{P}>$, **P** an element of order $q$ of **EC,**
$x$: **private** key $\mathbf{Y} = x.\mathbf{P}$: **public** key

Signing $m$:

- choose $k \in \mathbb{Z}_q$
- compute $\mathbf{R} = k.\mathbf{P}$
- compute $r = \mathit{first\text{-}coordinate}(\mathbf{R}) = f(\mathbf{R})$
- compute $e = H(m)$, $s = (e + xr)/k \bmod q$

$\sigma = (r,s)$

Verifying $(m,r,s)$: first $0 < r,\ s < q$

- compute $\mathbf{R'} = e\,s^{-1}.\mathbf{P} + r\,s^{-1}.\mathbf{Y}$

test if $r = f(\mathbf{R'})$

# Duplicate signatures for ECDSA

- Perform key generation as follows:
  - compute $h_1$ = H($m_1$), $h_2$ = H($m_2$)
  - choose $k \in \mathbf{Z}_q$ and compute $r = f\,(k.\mathbf{P})$
  - set **private** key to $x = -(h_1 + h_2) / 2r \bmod q$
  - set $s = (h_1 + x\,r) / k = -(h_2 + x\,r) / k \bmod q$
- Interpretation:
  - ECDSA is not broken
  - duplicate signatures reveal secret key
  - to eliminate duplicates need to tweak ECDSA



# Malleability of ECDSA

- In ECDSA $r = \textit{first-coordinate}(\mathbf{R}) = f(\mathbf{R}) = x_{\mathbf{R}}$
  Thus $f\,(-\mathbf{R}) = f\,(\mathbf{R})$
  Given a valid signature $(m,r,s)$,
  one obtains another as $(m,r,-s \bmod q)$
  This is exactly malleability
- Interpretation:
  - ECDSA is not broken
  - to eliminate malleability need to tweak ECDSA

# What does the proof tell?

- A security proof for ECDSA has been proposed in the **generic model**, where one gets access to elements of **G** through **encodings**
- Probabilities are computed by randomizing on encodings
- Theorem: *Non-malleability of ECDSA cannot be broken with probability significantly greater than* $5(n+1)(n+q_s+1)/q$

($q_s$ # of signing queries, $n$ # of group operations)



# In other words…

- The security proof "proves" a property that **does not hold** for the actual scheme
- Interpretation:
  - **EC** groups are **not generic** (they have automorphisms)
  - either change the model…
  - or tweak the scheme

# Conclusions (1)

- We have shown several flaws in applying proof methodologies to signature schemes
- They **are not mathematical errors** but misconceptions on the security model



# Conclusions (2)

- We have shown possible variants to the usual definition of security based on Existential Forgery and CMA,
  - either weaker (the SO-CMA scenario)
  - or stronger (requesting non-malleability)
- We believe that the strongest possible requirement should be adopted
- This would imply tweaks for ESIGN and ECDSA