RSA and Rabin Signatures
Signcrypotion

Alessandro Tortelli
26-04-06
Overview

• Introduction
• Probabilistic Signature Scheme PSS
• PSS with message recovery
• Signcryption
  – CSC1
  – RSA-TBOS
RSA and Rabin textbook signatures

• Textbook RSA and Rabin signatures are deterministic algorithms:
  – Given
    • (sk, pk) a key pair
    • M message
      Signature is uniquely determined by (sk, pk) and M

• Undesirable property
  – Adaptive chosen message attack permits Malice to obtain two different square roots of a chosen message and thereby factor the modulus (§10.4.5)

• Solution: probabilistic approach
Signatures with Randomized Padding

- Bellare and Rogaway initiate the work of signing with RSA and Rabin in a probabilistic method

**Probabilistic Signature Scheme PSS**

- PSS is a randomized padding-based fit-for-application digital signature scheme for the RSA and Rabin functions
- Similarities with the RSA-OAEP scheme even if:
  - OAEP encryption procedure makes use of the one-way part of the RSA function
  - PSS signature scheme uses the trapdoor part of the RSA function
PSS key parameters

- Let \((N, e, d, G, H, k_0, k_1) \leftarrow U \text{ Gen}(1^k)\)

  RSA key material: \((N, e)\) public
  \[d = e^{-1} \pmod{\phi(N)}\] private

- \(k = |N| = k_0 + k_1\) with \(2^{-k_0}\) and \(2^{-k_1}\) negligible quantities

- Signing and verifying algorithms make use of two hash functions:

  - \(H : \{0,1\}^* \mapsto \{0,1\}^{k_1}\) compressor
  - \(G : \{0,1\}^{k_1} \mapsto \{0,1\}^{k-k_1-1}\) generator

  G output is split in two sub-strings:
  - G1 has the first \(k_0\) bits
  - G2 has the remaining \(k-k_1-k_0-1\) bits
PSS Padding

\[ M \rightarrow r \rightarrow G1(w) \rightarrow H \rightarrow G1 \rightarrow 0 \rightarrow w \rightarrow r^* \rightarrow G2(w) \rightarrow G2 \]
PSS signature generation

\[ \text{SignPSS}(M, d, N) = \]

\[ r \leftarrow_U \{0,1\}^{k_0} \]

\[ w \leftarrow H(M \parallel r) \]

\[ r^* \leftarrow G_1(w) \oplus r \]

\[ y \leftarrow 0 \parallel w \parallel r^* \parallel G_2(w) \]

\[ return(y^d (\text{mod} N)) \]

K-bit string less than N, necessary in order for the modulo exponentiation to be conducted correctly
PSS signature verification

\[
\text{VerifyPSS}(M, U, e, N) = \]
\[
y \leftarrow U^e \pmod{N}
\]
\[
\text{Parse } y \text{ as } b \parallel w \parallel r^* \parallel \gamma
\]
\[
r \leftarrow r^* \oplus G1(w)
\]
\[
\begin{cases} 
\text{if } (H(M \parallel r)) = w \land G2(w) = \gamma \land b = 0) \\
\text{return(True)} \\
\text{else return(False)}
\end{cases}
\]
PSS Security

• As with the security proof of RSA-OAEP the security proof of RSA-PSS takes place in the random oracle model. Thus the security proof only provides heuristic evidence for security in the real world.

• Formal evidence is again derived from reduction to contradiction: breaking RSA-PSS will require roughly the same amount of work as it takes to solve the RSA problem \text{HARD PROBLEM}!
  – forgery -> full inversion in one go -> exact security

• RSA-PSS is existentially unforgeable against adaptive chosen-message attacks in the random oracle model under the assumption that the RSA problem is intractable
Signing with Message Recovery
PSS-R

- Main idea: a padding-signature scheme that also permits everybody to recover a signed message

- Original Scheme: Bellare and Rogaway

- Variation: Coron et al.
  - Is secure for signature usage (trapdoor part of RSA function) unforgeability under adaptive chosen-message attack
  - Is secure for encryption usage (one-way part of RSA function) unforgeability under IND-CCA2 mode
PSS-R Padding
Original of Bellare and Rogaway

\[ M \parallel r \]

\[ H \]

\[ w \]

\[ r \parallel M \]

\[ G \]

\[ s \]
PSS-R Padding
Variation of Coron et al.

\[ M \parallel r \]

\[ H \]

\[ G \]

\[ w \]

\[ s \]
**PSS-R key parameters**

- Let \((N, e, d, G, H, k_0, k_1) \leftarrow U \text{Gen}(1^k)\)

  RSA key material: \((N, e)\) public
  
  \[ d = e^{-1} \left( \text{mod } \phi(N) \right) \text{private} \]

- \(k = |N| = k_0 + k_1\) with \(2^{-k_0}\) and \(2^{-k_1}\) negligible quantities

- Signing and verifying algorithms make use of two hash functions:

  \[ G : \{0,1\}^{k_1} \mapsto \{0,1\}^{k-k_1-1} \]
  
  \[ H : \{0,1\}^{k-k_1-1} \mapsto \{0,1\}^{k_1} \]
PSS-R Signature Generation or Message Encryption

\[ PSS - R - Padding(M, x, N) = \]

1. \( r \leftarrow_U \{0,1\}^{k_0} \)
   \( w \leftarrow H(M \ || \ r) \)
   \( s \leftarrow G(w) \oplus (M \ || \ r) \)
   \( y \leftarrow (w \ || \ s) \)

2. \( \text{if}(y \geq N) \text{goto 1.} \)

3. \( \text{return}(y^x \pmod{N}) \)

\( x = d \) for signature generation
\( x = e \) for message encryption
PSS-R signature verification or decryption with Ciphertext validation

\[ PSS - R - UnPadding(U, x, N) = \]
\[ y \leftarrow U^x \pmod{N} \]
Parse \( y \) as \( w \parallel s \)
Parse \( G(w) \oplus s \) as \( M \parallel r \)

\[ \text{if } (H(M \parallel r) = w) \quad \text{return}(\text{True} \parallel M) \]
\[ \text{else} \quad \text{return}(\text{False} \parallel \text{Null}) \]
PSS-R Proof of security Encryption

- Proof of security is conceptually the same to that of RSA-OAEP
- A run of the attacker only causes a partial inversion
- Even running Malice twice, the reduction is far from tight (Number Field Sieve method works better if RSA modulus is less than 2048-bit)

\[ |w| > \frac{|N|}{2} \quad \rightarrow \quad |M| \leq \frac{|N|}{2} \quad \rightarrow \quad |M| \leq \frac{|N|}{2} - k_0 \]

Rahter low bandwith for message recovery
PSS-R Proof of security Signature

• Proof of security is conceptually the same to that of RSA-PPS
• Successful forgery of a signature can lead to full inversion of RSA function in one go
• It suffices for $k_0$ and $k_1$ to have size with $2^{k_0}$ and $2^{k_1}$ being negligible

$|M| = k - k_0 - k_1$
Signcryption

• Common practice:
  digital signature and then data encryption
  \[\text{Message expansion rate} \]
  \[\text{Computational time spent} \]

• Signcryption: is a public key primitive to achieve the combined functionality of digital signature and encryption
  – Zheng’s Signcryption Scheme: SCS1 (ElGamal)
  – Malone-Lee and Mao: Two Birds One Stone TBOS (RSA)
SCS1 parameters setup

• Public Parameters
  – p a large prime
  – q a large prime factor of p-1 (q||(p-1))
  – g an element of Zp* of order q
  – H: a oneway hash function
  – Setup a symmetric encryption algorithm $E$
SCS1 keys setup

- **Alice’s keys**
  - $x_a$: Alice’s private key, $x_a \in \mathbb{Z}_q^*$
  - $y_a$: Alice’s public key, $y_a = g^{x_a} \mod p$

- **Bob’s keys:**
  - $x_b$: Bob’s private key, $x_b \in \mathbb{Z}_q^*$
  - $y_b$: Bob’s public key, $y_b = g^{x_b} \mod p$
SCS1 Signcryption

To send to Bob $M$, Alice performs:

1. Pick $u$ randomly from $[1 \ q]$, computes $K = y_b^u \mod p$
   split $K$ into $K1$ and $K2$ of appropriate lengths
2. $ e \leftarrow H(K_2, M) $
3. $ s \leftarrow u(e + x_a)^{-1} \mod q $
4. $ c \leftarrow e_{k1}(M) $
5. Send to Bob the signcrypted text $(c, e, s)$
SCS1 Unsigncryption

• Recived \((c,e,s)\) from Alice, Bob performs:
  1. Recover \(K\) from \(e,s,g,p,y_a\) and \(x_b\):
     \[
     K \leftarrow (g^{e} y_a)^{sx_b} \mod p
     \]
  2. Split \(K\) into \(K_1\) and \(K_2\)
  3. \(M \leftarrow D_{K_1}(c)\)
  4. Accept \(M\) as a valid message originated from Alice only if:
     \[
     e = H(K_2, M)
     \]
SCS1 Efficiency/1

- **Computation:**
  - **Sygncryption:**
    - One modulo exponentiation
    - One hashing
    - One symmetric encryption
  - **Unsigncryption**
    - Similar amount of computation if \((g^e y_a)^{s_b}\) is rewritten to \(g^{esx_b} y_a^{sx_b}\) and computed using the “Product of Exponentiations Algorithm”. Otherwise it needs two modulo exponentiations.
SCS1 Efficiency/2

- **Communication bandwidth:**
  - Symmetric encryption doesn’t cause data expansion
  - Message signcrypted + $2|q|$ bits
    (same bandwidth for transmitting a signature and signed message in the ElGamal-family signature)
  - Suitable for sending bulk volume of data efficiently (for example using a block cipher with the CBC mode of operation)
SCS1 Security

• SCS1 is essentially a triplet ElGamal signature with a recoverable commitment
  unforgeability of signature under adaptive chosen-message attack
• Zheng has not given a reductionist proof on the IND-CCA2 security
• Perhaps only the intended receiver is able to recover the commitment value K, under adaptive chosen-ciphertext attack
SCS1 Non-repudiation

- i.e. a principal cannot deny the authorship of a message.
- In Zheng’s scheme, verification of a (triplet) signature requires recovery of the commitment $K$ and the recovery needs to use the receiver’s private key DRAWBACK!
- Third party’s arbitration cannot be done!
  - Bob can conduct a Zero Knowledge Proof to convince the arbitrator (tricky)
Two Birds One Stone

- Main idea: “double-wraps”
  - Alice first signs a message by “wrapping” it inside the trapdoor part of her own RSA function
  - Then encrypts the signature by further “wrapping” it inside the one-way part of the RSA function of an intended receiver (Bob)

\[
(N_A, e_A) \quad (N_A, d_A) \quad \text{Alice’s RSA public, private key material}
\]

\[
(N_B, e_B) \quad (N_B, d_B) \quad \text{Bob’s public, private key material}
\]

\[
M^{d_A} \pmod{N_A} \pmod{N_B}
\]
RSA-TBOS observations

• Alice’s RSA modulus may be larger than Bob’s one same moduli size
• In general a message is wrapped after the message has been processed with a randomized padding scheme
• If an “inner wrapping” result exceeds the modulus for an “outer wrapping”? 
  
  sender chops of the most significant bit
  receiver uses trial-and-error test to put it back
RSA-TBOS Key Parameters

- Let $k$ an even positive integer
- Let:
  
  $(N_A, e_A) (N_A, d_A)$  Alice’s RSA public, private key material
  $(N_B, e_B) (N_B, d_B)$  Bob’s public, private key material

  satisfying $|N_A| = |N_B| = k$

- Signing and verifying algorithms make use of two hash functions:
  
  $H : \{0,1\}^{n+k_0} \mapsto \{0,1\}^{k_1}$
  $G : \{0,1\}^{k_1} \mapsto \{0,1\}^{n+k_0}$

  Where $k = n + k_0 + k_1$ with $2^{-k_0}$ and $2^{-k_1}$ negligible quantities
RSA-TBOS Signcryption

When Alice signcrypts a message $M \in \{0,1\}^n$ for Bob, she performs:

1. $r \leftarrow_U \{0,1\}^k$
2. $w \leftarrow H(M \parallel r)$
3. $s \leftarrow g(w) \oplus (M \parallel r)$
4. if $(s \parallel w > N_A)$ goto (1.)
5. $c' \leftarrow (s \parallel w)^{d_A} \pmod{N_A}$
6. if $(c' > N_B)$, $c' \leftarrow c' - 2^{k-1}$
7. $c \leftarrow c'^{e_B} \pmod{N_B}$
8. Send $c$ to Bob
RSA-TBOS Unsigncryption/1

• When Bob unsigncrypts a cryptogram $c$ from Alice, he performs:

1. $c' \leftarrow c^{d_B} \pmod{N_B}$
2. if $(c' > N_A)$, reject
3. $\mu \leftarrow c'^{e_A} \pmod{N_A}$
4. Parse $(\mu)$ as $(s \parallel w)$
5. $M \parallel r \leftarrow G(w) \oplus s$
6. if $(H(M \parallel r) = w)$, return $(M)$
7. \( c' \leftarrow c' + 2^{k-1} \)
8. \( \text{if } (c' > N_A), \text{reject} \)
9. \( \mu \leftarrow c'^{e_A} \pmod{N_A} \)
10. \( \text{Parse}(\mu)as(s \parallel w) \)
11. \( M \parallel r \leftarrow G(w) \oplus s \)
12. \( \text{if } (w) \neq H(M \parallel r), \text{reject} \)
13. Return \( M \)
RSA-TBOS features

✓ Non-repudiation is very simple
  o The receiver of a signcryption follows the unsigncryption procedure up until stage 2, $c'$ may then be given to a third party who can verify its validity

✓ Message confidentiality under the IND-CCA2

✓ Signature unforgeability under the chosen message attack

❖ Rather low message bandwidth for message recovery due to the application of the RSA-PSS-R padding scheme