Status of Draft ANSI X9.44 (& More)

Burt Kaliski and Jakob Jonsson
RSA Laboratories
NIST Key Management Workshop
November 1–2, 2001
(Rev. November 6, 2001)

Outline

• ANSI X9.44 update
• Security proof background
• Scheme details
ANSI X9.44

- Key establishment schemes based on the integer factorization problem
  - Key transport and key agreement
  - RSA algorithm; also Rabin-Williams?
- Companion to ANSI X9.31 for signatures, ANSI X9.42/.63 for discrete logarithm / elliptic curve problem
- Along with other X9 documents, basis for NIST key management scheme FIPS

Design Choices

- Encryption schemes
  - Primitives and “encoding methods”
- Key transport schemes
- Key agreement schemes
Primary Methods

• PKCS #1 v1.5 encryption
  – 1991, no security proof, widely deployed

• RSA-OAEP
  – 1994, loose security proof, in some standards, not deployed

• RSA-KEM, et al.
  – 2001 (and previous), tight security proofs, brand new

Project Evolution

• RSA-OAEP in drafts through May 2001

• PKCS #1 v1.5 added in June 2001 to reflect practice, esp. SSL/TLS, but not for use with AES

• TLS working group decides in August 2001 to use PKCS #1 v1.5 with AES
  – NIST draft guideline reflects decision (TLS_RSA_WITH_AES_128_CBC_SHA)
Current Content

- Encryption schemes:
  - PKCS #1 v1.5
  - RSA-OAEP
- Key transport schemes:
  - One-pass with one public key
    - reflects S/MIME “recipientInfo”
- Key agreement schemes:
  - Multiple-pass with one public key, key confirmation
    - reflects SSL/TLS handshake

Related Efforts

<table>
<thead>
<tr>
<th></th>
<th>PKCS #1 v2.0</th>
<th>PKCS #1 v1.5</th>
<th>RSA-OAEP</th>
<th>RSA-KEM</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKCS #1 v2.0</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEEE Std 1363-2000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESSIE Phase 2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRYPTREC Eval.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO 18033-2 Draft</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>RSA-OAEP+</td>
</tr>
</tbody>
</table>
Outline

• ANSI X9.44 update
• Security proof background
• Scheme details

The Adversary

• An adversary is an algorithm that tries to break a cryptographic scheme, i.e., solve a problem that undermines the security of the scheme.
• Examples of such problems:
  – Find the plaintext corresponding to a ciphertext in an encryption scheme
  – Find the underlying secret key used to encrypt messages
  – Find the inverse of an element with respect to a function that is assumed to be one-way
  – Given a one-way function, find two elements with the same image (a collision)
Goals of Adversary

For asymmetric schemes, two kinds of security goals are normally considered:

• Indistinguishability of encryptions (IND)
  – Given two messages and the encryption of one of the messages (the target ciphertext), it is hard to determine which message is encrypted

• Non-malleability (NM)
  – Given a target ciphertext \( y \), it is hard to find another ciphertext \( y' \) such that the corresponding plaintexts are "meaningfully related"

The Strength of the Adversary

Depending on whether the adversary has access to an oracle performing private key operations, we obtain three basic levels of adversary strength:

• Chosen Plaintext Attack (CPA, offline attack)
  – The adversary can only encrypt messages

• Non-adaptive Chosen Ciphertext Attack (CCA1)
  – The adversary has access to a decryption oracle until, but not after, it is given the target ciphertext

• Adaptive Chosen Ciphertext Attack (CCA2)
  – The adversary has unlimited access to a decryption oracle, except that the oracle rejects the target ciphertext
    • The CCA2 model is very general – in practice, adversaries are much weaker than a full-strength CCA2 adversary
    • Yet, many adversaries are too strong to fit into CCA1
Six Attack Models

The implications between the attack models are as follows:

NM-CCA2 \leftrightarrow IND-CCA2

NM-CCA1

NM-CPA \leftrightarrow IND-CCA1

IND-CPA

Security in stronger model

Security in weaker model

Security Arguments

Security arguments can be divided into different categories, ranging from the strongest to the weakest:

- **Existence of stringent security proof**
  - We can *prove* that the scheme is secure under certain assumptions

- **Heuristic security arguments**
  - We have no proof of security, but we can give evidence that the scheme is hard to break
  - Security claims on symmetric ciphers and cryptographic hash functions belong typically to this category

- **Ad hoc arguments**
  - “The scheme is secure because there is no known attack”
Assumptions

Given a stringent security proof, there are a variety of possible assumptions on the underlying components, ranging from the strongest to the weakest:

- **No assumptions are needed**
  - The security proof requires no nontrivial assumptions
- **A certain mathematical problem is hard**
  - To break the scheme, you must solve the underlying problem
- **Some components are assumed to have “ideal” properties**
  - Examples: Random oracle model; Generic group model
- **Unconventional restrictions are put on the adversary**
  - The adversary is prohibited from performing certain operations

Hard Problem

- A typical security proof for an asymmetric encryption scheme ES translates a successful attack into a solution to an underlying hard problem $P$ (e.g., the RSA problem).
- Typical assumption:
  
  $P$ cannot be solved with probability $\varepsilon$ within time $t$

  Desired consequence:
  
  $ES$ cannot be broken with probability $\varepsilon^*$ within time $t^*$, where

  $t^* / t$ is as large as possible;
  
  $\varepsilon^* / \varepsilon$ is as small as possible.

- The better $t^*$ and $\varepsilon^*$, the tighter security proof
- The stronger attack model (e.g., CCA2 instead of CCA1), the smaller $t^*$ and the larger $\varepsilon^*$ (if proof even exists)
Diagram of Provable Security

- \( t \) = Complexity of most efficient solution to hard problem.
- \( t^* \) = Provable lower bound for encryption scheme in terms of \( t \).

Quality of Security Reduction

- For a security proof to apply to practical parameters, we typically need
  \[
  t^* = t \\
  \epsilon^* = \epsilon
  \]
- Proofs tend to have loose reductions that give useless security guarantees in practice
- Yet, the very existence of a security proof with bounds polynomial in \( t \) indicates that the algorithm design is sound
  - An attack is translated into one solution to the underlying problem – not necessarily the most efficient solution
  - The derived solution uses the adversary only as a black box, which may leave room for further optimizations
Summary

• Four parameters need to be taken into account when analyzing a security proof:
  • The challenge for the adversary
    – IND, NM, ...
  • Strength of the adversary
    – CPA, CCA1, CPA2, ...
  • Assumptions on the underlying primitives
    – Hard mathematical problem, ideal components, ...
  • Quality of security reduction (in case there are underlying nontrivial assumptions)
    – Tight, adequate, loose

Ideal Properties of a Proof

• The challenge for the adversary should be as easy as possible
• The adversary should be as strong as possible
• The assumptions should be as weak as possible
• Quality of security reduction should be as good as possible
Security Proof Template

Assume **ASSUMPTIONS** on underlying primitive(s)
Then SCHEME is secure in **MODEL**
against a CHALLENGE – ATTACK adversary
with TIME bound and SUCCESS bound
given ADDITIONAL constraints

**Example**
Assume impossible to invert RSA with prob. $\epsilon$ within time $t$
Then RSA-OAEP is secure in the random oracle model
against an IND – CCA2 adversary
with running time at most $t/2 - O(q^2)$ and
given at most $q$ oracle queries

desired success probability at least $4\sqrt{\epsilon}$

Outline

• ANSI X9.44 update
• Security proof background
• Scheme details
Key Establishment Schemes

• The goal in our setting is one of the following:
  – To transport a key from one entity to another (key transport)
  – To enable two (or more) entities to agree on a key (key agreement)

• Focus on two common schemes:
  – Key transport: Encrypt key with recipient’s public key
  – Key agreement: Encrypt key material with recipient’s public key; derive key from key material, nonces

• Security depends on underlying encryption scheme

Generic Key Transport Model

**Generic Key Transport Model**

**SENDER**

- `SymK` → `E`
- `Optional parameters` -> `Wrapped SymK`
- `Wrapped SymK` → `D`
- `SymK` ← `D`

**RECIPIENT**

- `Public Key` ← `E`
- `Private Key` ← `D`
Generic Key Agreement Model

CLIENT \[\text{PreSec} \rightarrow \text{E} \rightarrow \text{D} \rightarrow \text{PreSec}\]

SERVER \[\text{PreSec} \rightarrow \text{D} \rightarrow \text{PreSec}\]

Secret \[\text{H} \rightarrow \text{E} \rightarrow \text{D} \rightarrow \text{H} \rightarrow \text{Secret}\]

KDF \[\text{N}_1 \rightarrow \text{N}_2 \rightarrow \text{N}_1 \rightarrow \text{N}_2 \rightarrow \text{Secret}\]

Public Key \[\text{T}_1 \rightarrow \text{T}_2 \rightarrow \text{T}_1 \rightarrow \text{T}_2 \rightarrow \text{Secret}\]

Private Key \[\text{N}_1 \rightarrow \text{N}_2 \rightarrow \text{N}_1 \rightarrow \text{N}_2 \rightarrow \text{Secret}\]

Wrapped PreSec

Security Requirements

- For generic key transport, underlying encryption scheme should (ideally) be IND-CCA2
  - Wrapped SymK should not reveal information about SymK, given access to decryption oracle at other points

- For generic key agreement, underlying “key encapsulation” should be IND-CCA2
  - (Wrapped PreSec, T₁) should not reveal information about Secret, given access to decryption oracle at other points
  - Freshness, etc. are also important
Three RSA-Based Encryption Schemes

- PKCS #1 v1.5 RSA encryption
- RSA-OAEP
- RSA-KEM (“simple RSA”) + DEM

RSA-PKCS #1 v1.5

- Introduced in 1991 in PKCS #1
- *De facto* standard for RSA encryption and key transport
  - Appears in protocols such as TLS
- No security proof exists
  - Yet, no fatal attack against the scheme so far
RSA-PKCS #1 v1.5 Encryption

```
Padding

00 02 Padding 00

EM
```

RSA-PKCS #1 v1.5 Analysis

- Attack against low-exponent RSA when very long messages are encrypted
  - Attack applies if large parts of a plaintext is known or if similar messages are encrypted with the same public key
  - Mounted by Coppersmith et al. (1996); improvements by Coron et al. (2000)
  - Restrictions on the size of the plaintext help thwart attack
  - Not an issue in key agreement protocols

- Chosen ciphertext attack (“Million Message Attack”)
  - Requires a decryption oracle that reports whether a given ciphertext is valid or not
  - For a 1024-bit modulus, the attack requires about one million decryption queries
  - Mounted by Bleichenbacher (1998)
  - Attack is thwarted if ciphertext validity is not revealed, as in TLS
RSA-OAEP

- Asymmetric encryption scheme combining RSA with the OAEP encoding method
- OAEP was invented by Mihir Bellare and Phillip Rogaway in 1994
  - Additional enhancements by Don B. Johnson and Stephen M. Matyas in 1996
- Already widely adopted in standards
  - IEEE Std 1363-2000
  - ANSI X9.44 draft
  - PKCS #1 v2.0 and v2.1 draft

RSA-OAEP Encryption
RSA-OAEP Security

- RSA-OAEP is provably secure against IND-CCA2 in the random oracle model
  - Fujisaki, Okamoto, Pointcheval, and Stern (2000)
- Assume that the following is true:
  - The RSA encryption primitive cannot be inverted with probability $\epsilon$ within time $t$
- Then the following holds:
  - RSA-OAEP cannot be broken with prob. $\epsilon^*$ within time $t^*$, where
    $$\epsilon^* = 4\sqrt{\epsilon}$$
    $$t^* = t / 2 - O(q^2)$$
  ($q$ is the number of oracle queries)
- Unfortunately, the reduction is not tight

More on RSA-OAEP

- Bellare and Rogaway proved that RSA-OAEP is IND-CCA1 secure and conjectured IND-CCA2 security
- Shoup observed that a general IND-CCA2 proof for OAEP combined with any trapdoor function cannot be obtained
  - In general, the security of $f$-OAEP can only be related to the hardness of partially invert the underlying trapdoor function $f$
- Fujisaki, Okamoto, Pointcheval, and Stern demonstrated that the specific combination RSA-OAEP is IND-CCA2 secure
  - Unfortunately, security bounds are weaker than in the Bellare-Rogaway IND-CCA1 proof
RSA-KEM+DEM

- KEM = Key Encapsulation Mechanism
- DEM = Data Encapsulation Mechanism
- Construction goes back (at least) to Zheng and Seberry in 1992 and Bellare and Rogaway in 1993. Further development by Victor Shoup
  - RSA-REACT is a variant by Okamoto and Pointcheval
- RSA-KEM ("Simple RSA") generates a random integer $r$, derives a symmetric encryption key from $r$ via a key derivation function (KDF), and encrypts $r$ with RSA
- DEM encrypts a message $M$ with (e.g.) AES using the derived key
  - DEM can be combined with a keyed MAC of $M$, where the key is derived from $r$. The combination is denoted DEM1
  - If $M$ is key material, DEM can be AES key wrap

RSA-KEM Encryption

Diagram:
- $r$ is generated
- $r$ is fed into a key derivation function (KDF)
- The derived key is fed into a Data Encapsulation Mechanism (DEM)
- The output of the DEM is $C_0$ and $C_1$
- $M$ is the message
- The Public Key is used to encrypt a Random Integer $r$ with RSA
- The recipient must have the Private Key to decrypt $r$.
RSA-KEM Key Wrap

- RSAEP
- Public Key
- $r$
- KDF
- Derived key
- AES keywrap
- $C_0$
- Wrapped AES key
- recipient

Note: KDF can be based on AES via standard block-cipher to hash construction, so only RSA+AES needed

RSA-KEM Security

- RSA-KEM has a tight security, given the random oracle assumption on the KDF; we have
  \[ e^* = \varepsilon; \]
  \[ t^* = t - O(q) \]
- Reduction is linear in terms of the number of random oracle queries
- Security proof can be extended to RSA-KEM+DEM1 with the security expressed tightly in terms of the hardness of RSA and the security of the symmetric encryption and MAC algorithms
PKCS #1 v1.5 as a KEM

- RSA-KEM “encapsulates” keys as
  - \( K = KDF(r), \ c = f_{RSA}(r), \ r \ \text{random} \)

- PKCS #1 v1.5 (P1) can do so as
  - \( K = KDF(r), \ y = f_{P1}(r), \ c = (y, H(y,r), \ r \ \text{random} \)

- Claim: P1-KEM has tight security under the “Gap-P1” assumption
  - Hard to invert \( f_{P1} \) given a P1 “decision” oracle
    - Decision oracle indicates whether \( (y,r) \) is a valid P1 pair, i.e., \( y = f_{P1}(r) \)
    - TLS handshake using PKCS #1 v1.5 actually based on P1-KEM — so has tight security proof
      - \( K \ \text{derived from Secret,} \ y = \text{Wrapped PreSec,} \ H = T_1, \ r = \text{PreSec} \)

---

Generic Key Agreement Model

CLIENT

\( PreSec \)

\( KDF \)

\( Secret \)

\( PreSec \)

SERVER

\( N_1 \)

\( N_2 \)

\( E \)

\( D \)

\( T_1 \)

\( T_2 \)

\( H \)

\( H \)

\( \text{Wrapped PreSec} \)

\( “C”|N_1|N_2 \)

\( “S”|N_1|N_2 \)
Conclusion

- ANSI X9.44 draft moving along to guide and reflect practice
- Goal: consider what's in use, what can be proved
- RSA-KEM “key encapsulation” an alternate approach, after PKCS #1 v1.5, RSA-OAEP
- New security claims for PKCS #1 v1.5 key encapsulation, as in TLS

Backup Slides
Indistinguishability (IND)

- Intuition:
  - Given two messages and the encryption of one of the messages (the target ciphertext), it is hard to determine which message is encrypted.

- The IND adversary works in two steps:
  - After step 1, the adversary outputs two messages $x_0, x_1$.
  - Let $b = 0$ or $1$ with equal probability. Form a ciphertext $y$ by encrypting $x_b$ and give $y$ to the adversary.
  - After step 2, the adversary outputs a bit $b'$ that she believes equals $b$.
  - The adversary is successful if $b = b'$.
    - This means that she is able to distinguish between encryptions of $x_0$ and $x_1$.

Non-Malleability (NM)

- Intuition:
  - Given a target ciphertext $y$, it is hard to find another ciphertext $y'$ such that the corresponding plaintexts are "meaningfully related".

- The NM adversary works in two steps:
  - After step 1, the adversary outputs two messages $x_0', x_1'$.
  - Let $b = 0$ or $1$ with equal probability. Form a ciphertext $y$ by encrypting $x_b$ and give $y$ to the adversary.
  - After step 2, the adversary outputs a binary relation $R$ and a ciphertext $y'$.
  - Let $x'$ be the decryption of $y'$. The adversary is successful if $R(x', x_0')$ is true and $R(x', x_{1-b})$ is false.
Let A be an IND adversary. Define an NM adversary B as follows:

- After step 1, A outputs two messages \( x_0, x_1 \)
  - B outputs the same messages
- Let \( b = 0 \) or \( 1 \) with equal probability. Form a ciphertext \( y \) by encrypting \( x_b \) and give \( y \) to B
  - B passes \( y \) on to A
- After step 2, A outputs a bit \( b' \)
  - B forms \( x' = x_b + 1 \), encrypts \( x' \) to the ciphertext \( y' \), and outputs \((y, R)\) where \( R(u, v) \) is true if \( u = v + 1 \)

B is successful if A is successful:
- If \( b = b' \), then \( x' = x_b + 1 \) and \( x' = x_{1-b} + 1 \)

**IND and NM Example**

In “Pure RSA”, a plaintext \( x \) is encrypted as \( y = x^e \pmod{N} \)

Pure RSA does not satisfy the IND or NM criteria:

- NM is violated: Given a ciphertext \( y \), the ciphertext \( y' = yk^e \pmod{N} \) has the property that the corresponding plaintexts \( x \) and \( x' \) satisfy \( x' = xk \pmod{N} \)
  - This observation exploits the underlying mathematical structure of RSA
- IND is violated: It is easily checked whether or not a certain ciphertext is the encryption of a certain message.
  - This is true for any deterministic scheme and also translates into an NM attack
- Conclusion: RSA in itself does not provide any security
  - Yet, it may well be useful as a component in a larger scheme!
NM-CCA2 $\iff$ IND-CCA2

- $\text{NM} \implies \text{IND}$ is always true.
- For the other implication, let $B$ be an NM-CCA2 adversary. Define an IND-CCA2 adversary $A$ as follows.
  - After step 1, $B$ outputs two messages $x_0, x_1$.
    - $A$ outputs the same messages.
  - Let $b = 0$ or $1$ with equal probability. Form a ciphertext $y$ by encrypting $x_b$ and give $y$ to $A$.
    - $A$ passes $y$ on to $B$.
  - After step 2, $B$ outputs a ciphertext $y'$ and a relation $R$.
    - $A$ sends $y'$ to the decryption oracle (this is only possible in CCA2, not in CCA1!) and obtains a plaintext $x'$.
      - If $R(x', x_0)$ is true, than $A$ outputs $x_0$. Else, $A$ outputs $x_1$.
  - $A$ is successful if $B$ is successful.

Random oracle model

- A random oracle assumption on a function $H : X \rightarrow Y$ means that an adversary cannot compute or even predict the value of $H(x)$ for any $x$:
  - To compute $H(x)$, the adversary sends $x$ to a random oracle.
  - The oracle responds with a value chosen at random (typically uniformly) from the set $Y$.
  - The chosen value is independent from earlier queries and responses.
- In practice, a fixed function $h$ cannot be interpreted as a random oracle.
  - Outputs are fixed, not random.
- However, the assumption is useful in that it restricts the model to generic attacks not exploiting the inner structure of $H$. 
A more realistic oracle model?

- Suppose $H$ is randomized; $H$ takes as input an element $x$, generates a random $r$, and outputs $y = H'(r, x)$ with $H'$ fixed.
- Introduce an inversion oracle that finds $r$ such that $y = H'(r, x)$ for inputs $x$ and $y$.
- Drawback: If $H'$ is hard to invert in practice, the inversion oracle cannot be simulated, as opposed to random oracles.
  - The security can only be reduced to the hardness of solving an underlying problem given an inversion oracle.
- Possible advantages:
  - $H$ is a fixed (randomized) function even within the model.
  - The problem of inverting $H'$ might be “independent” from the underlying mathematical problem – solving one of the problems may not help in solving the other.
- Model introduced by Gennaro, Halevi, and Rabin.

Plaintext awareness

- A scheme with IND-CPA security is plaintext aware (PA) if an adversary cannot form a valid ciphertext without the corresponding plaintext being derivable from the oracle queries and responses.
  - The adversary has access to an encryption oracle and random oracles but no decryption oracle.
- PA implies IND-CCA2 security.
  - Decryption queries give no information since the adversary already “knows” the plaintext.
- Also, IND-CCA2 does not imply PA.
  - In an IND-secure scheme, the public key may leak a valid ciphertext without leaking the corresponding plaintext.
- PA makes sense only in the random oracle model.
  - In the standard model, the adversary can encrypt a plaintext and then “forget” it.
OAEP Parameters and Options

- **Encoding parameters**
  - Often empty, but other possibilities exist

- **Secure hash function**
  - Applied to the encoding parameters to produce a string \( pHash \)
  - Provides plaintext awareness

- **Mask generation function (MGF)**
  - Based on a secure hash function (preferably the one applied to the encoding parameters)
  - If the MGF is instantiated by a random oracle, the encoded message is (almost) uniformly random and independent from the original plaintext

RSA-OAEP+

- OAEP+ is an adaptation of OAEP introduced by Victor Shoup, replacing “\( pHash \)” with a hash of a string containing the plaintext and the seed
- OAEP+ can be combined with any secure trapdoor function, whereas OAEP is provably secure only with RSA and Rabin
- The security reduction for RSA-OAEP+ is better than that for RSA-OAEP; we have
  \[
  \epsilon^* = \epsilon; \\
  t^* = t - O(q^2)
  \]
- Yet, this is still not tight; the time bound is quadratic in the number of queries
SAEP(+) 

- SAEP is a padding method consisting of the first "Feistel round" of OAEP. SAEP+ is derived from OAEP+ in the same manner
  - SAEP introduced by Johnson and Matyas in the early 90s
  - SAEP+ designed by Boneh in 2000
- Rabin-SAEP+ has a tight reduction that is linear in the number of queries (i.e., $t^* = t - O(q)$)
  - Yet, Rabin schemes are vulnerable to implementation weaknesses that may leak the entire private key
- RSA-SAEP+ has a security reduction roughly equivalent to that of RSA-OAEP

RSA-KEM+DEM for Key Transport

- RSA-KEM produces a key that can be used to encrypt data
  - Suitable in some situations, but not always
  - Gives a ciphertext overhead of a multiple of 128 bits (in the case of AES) compared to e.g. RSA-OAEP when the message is small
  - Not appropriate in multiparty situations where the same data should be distributed to many entities
    - The same $r$ cannot be used more than once
- RSA-KEM+DEM can also be used to encrypt a previously generated key
  - Solves the multiparty problem
  - Fits nicely into existing protocols where the secret key is generated outside the PKE module
  - Yet, still gives ciphertext overhead compared to RSA-OAEP