Summary of Victor Shoup’s ISO Document on 
Public Key Encryption

Rachel Shipsey
Royal Holloway, University of London

March 22, 2001

NES/DOC/RHU/WP3/014/a

1 Submissions to the ISO Committee

In April 2000, the ISO committee put out a call for encryption algorithms. The call for contributions proposed four parts: general, asymmetric, block and stream ciphers. [?] and this document deal only with the asymmetric ciphers. The following public key algorithms were submitted. Those marked with a * have also been submitted to NESTIE.

EC-DH Schemes

- ECIES *
  A Diffie-Hellman type scheme based on the Computational Diffie-Hellman problem for elliptic curves. ECIES is not secure against adaptive chosen ciphertext attack.

- PSEC *
  A family of elliptic curve Diffie-Hellman based encryption schemes claimed to be secure in the random oracle model under different assumptions. PSEC-1, -2 and -3 rely on the hardness of the DDH, CDH and gap-CDH problems respectively. Shoup argues that the security claims for PSEC-1 and PSEC-2 are unjustified. PSEC-3 is very similar to the ECIES submission.

DH Schemes

- ACE-Encrypt *
  A Diffie-Hellman scheme based on the DDH problem for a subgroup of \( \mathbb{Z}_p^* \). It can be proven secure against adaptive ciphertext attack without reference to the random oracle model, but is not as efficient as PSEC.
Factorisation Schemes

- **RSA-OAEP**
  A well established RSA scheme using the OAEP padding scheme. RSA-OAEP is secure in the random oracle model, however the security reduction is highly inefficient. RSA-OAEP only encrypts messages of short length.

- **EPOC**
  A family of encryption schemes, EPOC-1, -2 and -3, based on factoring integers of the form \( n = p^2q \) and claimed to be secure in the random oracle model under various assumptions.

- **HIME**
  A family of encryption schemes based on factoring integers claimed to be secure in the random oracle model under various assumptions. One of the HIME schemes is basically RSA-OAEP with encryption exponent 2.

The purpose of \([7]\) is to briefly describe all of the public key submissions, discard those thought not to be worthy of further scrutiny and propose changes to those still under consideration.

2 The API Framework

Although all of the algorithms are based upon either factoring problems or Diffie-Hellman type problems, there are many differences between the schemes including:

- **Length of message to be encrypted** - some schemes allow for arbitrarily long messages whilst others can only cope with short messages;

- **Some schemes allow for additional data to be “non-malleably bound” to the ciphertext**;

- **Some schemes allow for cleartexts and ciphertexts to be processed as “streams” whilst others require more than one pass over the data**;

- **Some schemes have claims of “provable security” against adaptive chosen ciphertext attack. Of these security claims, some rely on the “random oracle” heuristic whilst others do not, and some of these claims have been proven invalid.**

In \([7]\) it is proposed that all algorithms adopted as standards should fit into an informal API framework. The purpose of this framework is to unify the schemes, making them more comparable with each other in terms of security and efficiency, and allowing for the efficient construction and implementation of...
a “hybrid” encryption scheme where a public key algorithm is used to encrypt a key which is then used in a symmetric key algorithm. In particular selected schemes should have the following properties:

- Diffie-Hellman based encryption schemes should be specified with respect to an abstract group which could be implemented in several ways, including as a subgroup of an elliptic curve group or as a subgroup of $\mathbb{Z}_p^*$. 
- Schemes should be able to process cleartexts of arbitrary length.
- Schemes should be “provably” secure against adaptive chosen ciphertext attack in some reasonable sense.
- Schemes should provide a fairly unique efficiency/security tradeoff.

2.1 Key Encapsulation Mechanisms

The API framework really requires that the algorithms are “key encapsulation mechanisms”, referred to as $KEM$. Briefly, a $KEM$ works just like a public-key encryption scheme, except that the encryption algorithm takes no input other than the recipient’s public key. A $KEM$ consists of three algorithms:

- A key generation algorithm $KEM.KeyGen()$ which outputs a public key/secret key pair $(PK, SK)$;
- An encryption algorithm $KEM.Encrypt(PK, format)$ which takes as input a public key $PK$ and a format value, and outputs a key/ciphertext pair $(K, C_0)$;
- A decryption algorithm $KEM.Decrypt(SK, C_0)$ which takes as input a secret key $SK$ and a ciphertext $C_0$, and outputs a key $K$.

Note that a “format” value specifies a particular way to format a ciphertext. It is assumed that there are only a small, constant number of formatting choices for any particular cryptosystem, and that for a given public key, there is always a default format value. For the cryptosystems described here, the only use of this parameter is in the case of elliptic curve schemes where the encryptor may have to decide whether or not to use compact point representation.

2.2 Hybrid Encryption

Given a $KEM$ as described above, a symmetric encryption scheme $SKE$, and a one-time message authentication code $MAC$, one can build a hybrid encryption scheme that is secure against chosen ciphertext attack as follows:

We require that $KEM.OutputKeyLen = SKE.KeyLen + MAC.KeyLen$. Then given a cleartext message $M$ with label $L$, the $KEM$ is run yielding
a key $K$ of length $SKE.KeyLen + MAC.KeyLen$, and a ciphertext $C_0$. $K$ is parsed as $K = k \parallel k'$ where $k$ is of length $SKE.KeyLen$ and $k'$ is of length $MAC.KeyLen$. Message $M$ is then encrypted using $SKE$ with key $k$ to obtain $C_1$. We then apply $MAC$ to $T = C_1 \parallel L \parallel \Pi_0 SP([L], 8)$ using key $k'$ to obtain $tag$. The entire ciphertext is $C = C_0 \parallel C_1 \parallel tag$.

Decryption using the hybrid scheme is as one would expect. If the underlying components are secure then the resulting hybrid scheme is secure against adaptive chosen ciphertext attack.

Note that a “label” is a byte string which is effectively bound to the ciphertext in a non-malleable way. It should be meaningful to the application using the encryption scheme and independent for the implementation of the encryption scheme. A good labeling mechanism should not unnecessarily increase the size of the ciphertext.

3 Algorithms Under Consideration By ISO

Of the schemes submitted to the ISO committee, EPOC and HIME have been rejected. In the case of EPOC it was felt that the underlying theory and implementation details have not been widely enough scrutinised, and that the efficiency/security tradeoff is not good in relation to the other schemes. In the case of HIME, too many details were missing from the submission and the security proofs are questionable. PSEC-1 has been discarded due to “significant security problems” and PSEC-3 is thought to be too similar to ECIES, a better established scheme, and has therefore been rejected. Of the remaining algorithms, all have been modified to a greater or lesser extent leaving the following 5 algorithms:

- ECIES!

A variant of the ECIES encryption scheme which is a $KEM$. If the $KEM$ is used in a hybrid scheme as described above with an elliptic curve used to define the group, then the resulting scheme is very similar to the original ECIES submission. However there are the following differences:

- The group used to specify ECIES’ is the abstract group and not necessarily elliptic curve based.
- In ECIES’, one of the two labels [SharedInfo1] and [SharedInfo2] has been dropped.
- ECIES is not secure in the non-malleable sense. This is remedied in ECIES’ by including a hash of $C_0$ (which is generated from the group generator and temporary random variable) in the mask-generating function.
• PSEC-2’
A variant of the PSEC-2 encryption scheme which is a KEM designed for use in a hybrid scheme as described above. PSEC-2’ is different in many details from the original submission. In particular:

– Although PSEC-2 is in fact a KEM with it’s own hybrid construction, PSEC-2’ uses a different construction which fits in with the required API framework. The hybrid construction proposed in the API facilitates the implementation of the encryption and decryption algorithms as filters, reading and writing their inputs and outputs as streams - for the original PSEC-2 construction this is not possible. The hybrid construction of PSEC-2 relies upon the random oracle for its proof of security whereas the hybrid construction proposed in the API does not.

– The ciphertexts and code for PSEC-2’ are slightly longer because an additional MAC tag is required. This is to facilitate “streaming” and to make PSEC-2’ conform with the other schemes.

– In PSEC-2’, the random seed $s$ is hashed before use in the mask-generating function leading to potentially more compact ciphertexts.

– There is no detailed proof of the claimed security of PSEC-2, and some doubts as to whether or not the claims are valid. Shoup gives a proof of security in the random oracle model for the KEM PSEC-2’.

– The length parameters given in PSEC-2 cause a security problem as the ciphertext may contain some bits of PEH (generated from temporary random key and public key) in the clear. This is remedied in PSEC-2’.

– PSEC-2’ works with any cyclic group, not just elliptic curve groups, and not just of prime order.

• ACE-Encrypt’
Several changes have been made to ACE-Encrypt so that the resulting scheme fits in with the API framework. The result is a KEM. The main differences between ACE-Encrypt and ACE-Encrypt’ are

– The algorithm has been generalised to work with an arbitrary abstract group, an arbitrary message authentication code and an arbitrary symmetric encryption algorithm.

– The hash functions used are the standard cryptographic hash function rather than the specialised hash functions (based on MARS) proposed in the ACE-Encrypt submission.

• RSA-OAEP+

RSA-OAEP can only be used to encrypt messages of a bounded length. Because of this, RSA-OAEP is really only useful as a KEM and this is a
waste of one of the main features of OAEP - namely its very good message expansion rate. To facilitate encryption of arbitrary length cleartexts, an extended message encoding scheme XEME-OAEP+ is used. The RSA encryption scheme using extended message encoding scheme XEME-OAEP+ is called RSA-OAEP+.

- Simple RSA
  Simple RSA is a KEM which can be used as an encryption scheme using the hybrid construction of section 2.2. RSA is used together with a mask-generation function to output a pair \((C, K)\) as described in section 2.1. The main advantages of this scheme are its simplicity and the fact that it yields a much more efficient security reduction compared to that for OAEP or OAEP+.

4 Comments and Questions

Is there any point in NESSIE spending a lot of time analysing algorithms such as EPOC which have already been rejected by ISO and are therefore very unlikely to be accepted as standards.

The Quartz, Flash and SFash algorithms submitted to NESSIE were not submitted to the ISO. However if they had been, they would probably have been rejected on the grounds that the underlying problems have not been well studied, and there are no proofs of security - in the random oracle model or otherwise. However these algorithms are designed for high speed implementations rather than for high security implementations. It is therefore difficult to make meaningful comparisons between algorithms such as Quartz and, for example, ACE-Encrypt.

Should we be considering NESSIE submissions as submitted or with modifications? Since Shoup has turned the algorithms into key encapsulation mechanisms, which were not asked for in the NESSIE call for submissions, we should presumably consider the original algorithms rather than the ISO versions. However in the cases when small changes are obviously necessary, for example changing ECIES so that it is non-malleable, should we be changing the algorithms? If so, how far can we go with such changes. Presumably it's not really up to NESSIE to correct the submitted algorithms.