Summary of Evaluation

1 Introduction

This document is a summary of the security evaluation performed as requested by IPA. The following schemes were reviewed:

1. **ACE-encrypt**
2. **ECAES**
3. **ECDHS**
4. **ECMQVS**
5. **EPOC**
6. **HDEF-ECDH**
7. **HIME1**
8. **HIME2**
9. **PSEC**

Actually, since each of EPOC and PSEC includes three versions, this is altogether thirteen schemes. Among those schemes

- four are public key schemes: EPOC-1, HIME1, HIME2 and PSEC-1
- six are hybrid schemes: ACE, ECAES, EPOC-2, EPOC-3, PSEC-2 and PSEC-3
- three are key-agreement schemes : ECDHS, ECMQVS and HDEF

In each case, we have briefly reviewed the cryptosytem, discussed the security level of the cryptographic primitive which underlies it, and provided a detailed security analysis. We have refrained from stating any comparative opinion, but we believe that the reports that we have written are conclusive enough to compare the schemes, taking

into account additional evaluations, that were not part of our assignment, notably related to efficiency.

In the present summary, we have tried to provide our conclusions in a synthetic manner. They appear in a table, on which we briefly comment. This table shows figures. These should not be taken as grades. They are just a convenient way to express our findings. Our aim was to check that each scheme was secure using the strongest notion currently considered, security against chosen-ciphertext attacks. In the case of key agreement schemes, we used the formal security models that recently appeared in the literature and discussed whether the schemes could provide forward secrecy. In all cases, we also tried to assess that the proposed range of parameters could guarantee security for the foreseeable future.

Security assumptions and proofs Modern cryptology tries to relate cryptographic schemes to the computational hardness of solving algorithmic problems such as factoring or the discrete logarithm problem. Progress related to these problems is well documented. Other problems that have been around for some time are the RSA problem, the Diffie-Hellman computational problem (and its elliptic curve version), and the discrete logarithm problem on elliptic curves. Ideally, one would like to produce a mathematical proof based on one of these problems. However, no scheme currently admits such proof. To overcome this difficulty, there are different ways:

- Either, one can use stronger computational assumptions. Examples are the decisonal Diffie-Hellman assumption DDH (and its variant over elliptic curves), or the p-subgroup problem. A further step is to make *ad hoc* assuptions or assumptions which do not relate to algorithms but to interactions, such as the recently proposed "gap" hypotheses.

- Or else, one can replace mathematical proofs by proofs using the random oracle model, where hash functions are considered as random functions.

In the table below, we have classified the various approaches taken by the proposals under review. Figure 3 means the combination of a mathematical proof based on the hardness of a non-classical problem such as DDH and a random oracle proof from a standard assumption. Figure 2 indicates a random oracle proof based on a standard assumption. For non-standard assumptions and strong heuristic security, we use 1. Zero indicates that we met a problem while attempting to formally prove the security of the current specification.

Confidence Security proofs are elaborate mathematical arguments. Our view is that such a proof should be cross-checked. During the time of our review, it happened that a proof that had appeared more than five years ago, was found incorrect and later

repaired. Here, the ideal situation is a complete security proof designed for a given submission, which we grade as 3. We use 2 for submissions with adequate references to the literature and 1 when no formal security argument is provided in the submission. We use 0 when we met a problem while attempting to piece together the submission and the references it provided. We do not qualify proofs for which we have met a earlier problem.

Quantitative estimates Most of the security proofs show a security loss: attacking the scheme results in breaking the underlying hard problem with increased running time or decreased success probability. We respectively use figures 3, 2, 1 to indicate whether the reduction is tight, loose or with almost no practical meaning when applied to concrete situations. We do not qualify estimates for proofs where we have met an earlier problem.

Size of parameters We use 2 or 1. The former indicates that the range of parameters allows long-term security (based on the state of the art). The latter guarantees short-term security (something like ten to fifteen years), without proposing a large enough range for future evolutions.

Efficiency Although this was not the subject of our report, we mention efficiency. This appears useful as an increased formal security usually entails a corresponding loss in terms of efficiency. We use figures from 3 to 1. The lower figure corresponds to the heavier computing load.

	security	confidence	estimates	parameters	efficency
ACE-encrypt	3	3	3	2	1
ECAES	1	2	1	2	3
ECDHS	1	1	2	2	3
ECMQVS	1	1	2	2	3
EPOC-1	0	-	-	1	2
EPOC-2	2	2	2	1	2
EPOC-3	1	2	3	1	3
HDEF-ECDH	0	-	-	2	3
HIME1	2	0	-	1	3
HIME2	2	0	-	1	3
PSEC-1	0	-	-	2	2
PSEC-2	2	2	2	2	2
PSEC-3	1	2	3	2	3