CONNOTECH Experts-conseils inc.

PEKE, Probabilistic Encryption Key Exchange, 10 Years Later,

Including the PEKEv1.25 Specifications

Thierry Moreau

Document Number C003449

2005/10/03

Abstract

This document revisits the PEKE (Probabilistic Encryption Key Exchange) cryptosystem and proposes the enhanced PEKEv1.25 that performs a hash computation on the original PEKE output in order to improve the security assurance and to broaden the field of use. For a key establishment application where only the server side publishes a long-term public key and can adequately protect the private key counterpart from implementation attacks, we claim that PEKE is unsurpassed in security and efficiency, among the finite field arithmetic cryptosystems (e.g. RSA and finite field Diffie-Hellman). We use an original definition for the type of key encapsulation service provided by PEKE, hoping that this abstract definition captures the characteristics of the protocol and usage context. However, we only suggest that related security proofs are encouraging for the security of PEKE.

Table of Contents

Document Revision History

C-Number	Date	Explanation
C003428	2005/06/16	Initial release
C003449	2005/10/03	Added Annex A and reference [40]

1. Introduction

In the field of public key cryptography, the foremost deployed schemes fall into four categories, based on the underlying “hard”problem and the type of arithmetic used in the implementation:

- the discrete logarithm systems using finite field arithmetic (derivatives of the Diffie-Hellman cryptosystem),
- the RSA cryptosystem using finite field arithmetic,
- the Rabin-Williams cryptosystem and derivatives, using finite field arithmetic, and
- the discrete logarithm systems using elliptic curve arithmetic.

This classification is influenced by the purpose of the present document. Namely, we omit the rich diversification of the discrete logarithm cryptosystem family in either type of arithmetic, and the distinction made between RSA and Rabin-Williams is introduced despite their common reliance on the difficulty of factoring integers.

Traditionally, the main public key cryptosystem applications are digital signatures and public key encryption, the later being used in *hybrid* cryptosystems. We find convenient to state a definition of *key encapsulation scheme* for some cryptosystems that can provide the public key portion of an hybrid cryptosystem, but unsuitable as for direct public key encryption.

In this document, we revisit the *Probabilistic Encryption Key Exchange* cryptosystem, PEKE, published in 1995 [1]. Since then, little attention was paid to this proposal. But we were puzzled by some security and operational properties that we didn't see in on-going public key developments. Our definition of key encapsulation scheme conveys the PEKE unique properties that we would like to present as a challenge, both for PEKE review, and the appraisal of alternate solutions. Quite pragmatically, the author's small organization promotes a key management solution called SAKEM (Secret Authentication Key Establishment Method) that uses the PEKE cryptosystem for its internal use [2]. We have been facing the following dilemma for a while: there is a business incentive to replace the little known PEKE by an industry recognized solution, but we postponed this move because we lack a compliant candidate key encapsulation scheme.

Usually, research communications in the field of cryptography propose theoretical

advances with the hope that it may be useful to some practitioners. This note takes the opposite route. It is written from a practitioner perspective, with open questions to the more mathematically-inclined researchers. In selecting a public key solution, a system designer may pick up a standard method that was decided by a reputable organization, and abide by it. We prefer a different, practitioner approach, i.e.

- to challenge the rationale and every single decision element of reputable organizations, and
- to look for a comprehensive rationale for a public key method selection.

In this approach, we feeds our quest for public key methods with the theoretical results that we finds enlightening.

Although our motivations were present for some time, this document preparation was triggered the recent advances in the research for "provable security," mainly [3] and [4].

2. Background Review

2.1 Early Cryptosystems

The discovery of public key cryptography is usually associated with the Diffie-Hellman scheme [5] and the RSA scheme [6]. Twenty five years later, the Diffie-Hellman scheme derivatives are practiced with various optimizations, most notably the elliptic curve cryptography, and the RSA primitive usage is quite different from what was originally expected, i.e. small public exponent, hashing for digital signatures and hybrid cryptosystems for encryption.

Shortly after the publication of Diffie-Hellman and RSA, the Rabin-Williams scheme [7], [8] was published. The Rabin-Williams was the first one with security support by number-theoretic proofs.

Probabilistic encryption is a refinement of public key encryption. It was defined in [9], and then refined in the Blum-Goldwasser probabilistic encryption publication [10]. It provides an increased level of security by preventing an eavesdropper from gathering *any* information from the ciphertext, e.g. even if the attacker can himself encrypt the exact same cleartext as the legitimate user, the two ciphertexts are completely unrelated. Although the original probabilistic encryption proposals are seldom used in practice, similar security characteristics are sometimes provided in security protocols using secret random values. The theory justifying probabilistic encryption remains as a foundation of current "provable security" research. The PEKE scheme is derived from the Blum-Goldwasser probabilistic encryption.

In every Rabin-Williams variants and in the Blum-Goldwasser probabilistic encryption, the elementary public key computation is the function $f(x)=x^2 \bmod N$ with N being the product

of two distinct prime numbers, each congruent to 3 modulo 4. The most detailed number-theoretic review of the $x^2 \bmod N$ function remains reference [11] where the security proofs are applied to the design of cryptography-strong pseudo-random number generators. We hereafter refer to the BBS pseudo-random sequence generator by the name of the authors of [11]. See also [12] for early work about the number of bits that can be extracted at each step of the BBS generator.

2.2 The Selection between RSA and Rabin-Williams

With the Rabin-Williams scheme, the system designer are more knowledgeable about the security properties than with any other cryptosystem. This knowledge includes both good news and bad news, the good news are the security equivalence with the difficulty of factoring large integers. The bad news is the Rabin-Williams vulnerability to chosen ciphertext attacks, which force the system designer to be very cautious about how elementary private key computations is implemented and accessed by users and their applications software.

Looking back in the years 1985-2001, it looks as if a significant portion of the system designer community preferred the uncertainties of the RSA scheme, for which the security foundation is conjectural. Also, the various RSA vulnerabilities were discovered as the RSA usage evolved. Nowadays, the prevailing low public exponent RSA digital signatures is very close to the original Rabin-Williams scheme: recommended public exponent values are $2^{16}+1$ and 3 for RSA, and 2 for Rabin-Williams. The rationale for recommending $2^{16}+1$ over 3 seems to be folklore, while the greatest advantage of avoiding the public exponent 2 might have been the economic value of the RSA patent in the United States. The RSA patent expired in 2000, but the patent economics played an important role in determining the schemes that became the "installed base." In spite of the RSA predominance, and mostly driven by European cryptographers, the public exponent value 2 was standardized (along with 3 and other small odd exponents) as early as 1991 ([13], [14]).

It is interesting to note that the academic Rabin-Williams scheme encompasses the public exponent 3 ([7], [15]). This particular scheme uses a type of public modulus N that is incompatible with the RSA computations and an exponent value 3, i.e. if the public modulus is $N=P*Q$, then RSA with a public exponent e requires that $gcd(e,(P-1)\times(Q-1))=1$, which is always false with the Rabin-Williams scheme. In other words, the public exponent 3 is shared by the RSA and Rabin-Williams schemes, but using different computation sequences and different theoretic support.

The Rabin-Williams cryptographic primitive has an intrinsic 4:1 ambiguity, i.e. exactly 4 values of x satisfy $x^2 \bmod N=y$ for a given "quadratic residue" y. Each of the Rabin-Williams scheme variants must accommodate this ambiguity, and the relevant details are frequently the

main differentiation among the Rabin-Williams variants. References in this category but not cited elsewhere in this document include [16], [17], [18], [19], [20], [21], [22], and [23]. We claim that PEKE [1] is more than yet another Rabin-Williams encryption scheme, because it derives from probabilistic encryption and it complies to our definition of key encapsulation.

2.3 Developments in complexity security proofs

Over the last 5 to 10 years, there has been a number of advances in "provable security" of public key cryptosystems ([24], [4], [25]). This work is fairly involved. It is based on number theory, computational complexity theory in its more applied aspects, and a few theoretical attack models. Despite their very specialized nature, these theoretical advances influence international standardization of cryptographic schemes ([26] and [27]).

The mere understanding of how the security proofs work is a non-trivial mental exercise. Here is an attempt to an intuitive explanation, with some simplifications. The early security proofs (e.g. [11]) were qualitative appreciations, e.g. there is *some* key size large enough to reduce an attacker's chance of success to a small probability, however small this probability. In contrast, recent research focuses on *quantified* security for various schemes including hash operations, coding with random salt fields, and other algorithmic parameters with explicit bit sizes. These are formal algorithmic steps in an hybrid cryptosystem, i.e. pre- or post-processing to the public key elementary operation. The quantified results assume that the adversary is powerful in the sense that it may request operations using the private key with a chosen ciphertext. The quantitative results state something like "it is *relatively that more likely* that the adversary breaks the scheme using chosen ciphertext operations than it is to solve the underlying difficult problem directly" (e.g. factoring large integers). From these relative probabilities and the state-of-the-art algorithms for the underlying difficult problem, some gurus come up with specific key size recommendations, see section 5.6 in [28] and section 7.2.2.3 in [25]. A "tight reduction" proof is one showing that it is *negligibly* more likely that the chosen ciphertext adversary breaks the scheme.

How useful are the provable security proofs in practice? For sure, a system designer can be confident that a scheme backed by such a proof , once reviewed by the cryptographic community, has no internal flaws. Widely deployed schemes that lack theoretical support might be questioned. Some variants of RSA seem to fall in this category. PEKE has not been reviewed by theoreticians, so its lack of tight reduction proof is currently not a caution signal.

These developments in security proofs brought tight reduction proofs for Rabin-Williams encryption, see [3], and for digital signatures, see [29]. The view that Rabin-Williams enjoy the best theoretical support is expressed in [4], where the author comment on [3] as follows:

> "It is ironic that after a quarter century we come full circle and return to Rabin encryption, which was the very first public-key system designed so as to have a reductionist security

property. Boneh shows that all it needed was some random padding and hashing and just one Feistel round to go from being totally insecure to totally secure against chosen-ciphertext attack."

With the original PEKE design, we took probabilistic encryption, turned it into a key encapsulation scheme and added some integrity padding. The 4:1 ambiguity of Rabin-Williams derivative is handled in PEKE as if the 4 solutions were indistinguishable. In [1], we suspected that the PEKE scheme might still be insecure against chosen ciphertext attacks, so we indicated usage precautions with the generated symmetric key. After encountering the reference [3], we are inclined to add a 1.25 hash compression on the symmetric key output, this time with confidence that the resulting PEKEv1.25 is resistant against chosen ciphertext attack, thus relaxing the usage precautions with a small performance penalty.

We whish we were able to provide security proofs for PEKEv1.25 with an extensive mathematical development that could be reviewed by academia. However, these very complex proofs use assumptions closely tied to the details of the encryption scheme. Thus, the proof in [3] for general purpose encryption is not readily converted into a proof supporting PEKE. This is not to say that the published results are of little use. On the contrary, by focusing on minute computational details (e.g. the precise arguments passed to a hash function), a proof provides confidence in a specific applications-oriented scheme.

2.4 Other Formal Security Rationales

Despite their strength, the computational complexity proofs are centered on the elementary cryptographic primitives, leaving aside many protocol issues. In our practice-oriented but intuitive search for theoretical support of public key solutions, we selected no academic reference in the protocol security modeling area.

As a practitioner of public key cryptography, we whish the theoretical support would extend further beyond primitives and protocols. We like the idea of "failure modes and effects analysis" (engineering acronym FMEA) used in the fields of critical industrial infrastructures, such as nuclear or aviation security. This involves an engineering study of consequences of a component failure in a complex system. In the field of information security, relevant component failures include e.g.

- a reduced entropy in a party's random number generator,
- the occasional use of a low entropy password where the system rules mandate a high entropy shared secret value, or
- the surreptitious replacement of a remote party's public key in a system configuration.

In these examples, the system's operation may proceed with the component impairment unnoticed, and the impaired party otherwise behaves as expected (the security protocol modeling typically assume that behaving parties are perfectly good and misbehaving parties can be very bad). Under a failure mode analysis, the security of a system is appraised under the various

operating conditions with various component impairments.

2.5 Implementation Attacks

Another aspect of cryptographic security is the *implementation attacks*. This encompasses passive interception of any logical or physical signals emanating from the computing device while it runs the cryptographic computations, e.g. when the secret parameters are handled in the CPU (Central Processor Unit). Active implementation attacks cover cases where an adversary attempts to trigger faults in the normal cryptographic computations so that the computing device handling of secrets is diverted from its normal sequence. Since computing devices are implementing mathematical abstractions in a very real world, albeit electromagnetic, implementation attacks are a concern, especially serious with small or cheap computing devices such as a smart-card or a personal computer lacking elementary protections against electromagnetic leakage of information. An implementation attack is described in [30], which discloses logical and timing-based attacks to pre- or post-processing on a current RSA implementation variant, without directly targeting the RSA private key.

The Rabin-Williams cryptosystem is not different from other public key schemes with respect to implementation attacks. However, the well known theoretical strength and weaknesses should facilitate the implementation of countermeasures against the compromising signal emanations, e.g. starting from a careful study of reference [11]. Generally, timing-based implementation attacks should be prevented by a software that runs every step of the computation, no matter the step at which the final outcome is determined. With any of the Rabin-Williams variants, the software portions that handle the 4:1 ambiguity should be carefully checked, without reluctance to insert processing *in*efficiencies that hide how the final outcome is determined.

Like any other vulnerability, implementation attacks are more critical to keys that are higher in the key hierarchy of a key management scheme, such as the signature key of a certification authority. For key encapsulation schemes, a similar key management hierarchy summit occurs if the key encapsulation transactions are used for the registration of long-term keys. Either way (digital signature for certification authority or key decapsulation for registering long-term keys with a central organization), the public key computation should be done in a dedicated secure cryptographic device for which adequate protection can be afforded. Such secure device architectural issues are covered in reference [31] for digital signature. Similar secure device architecture is deserved to protect the private key computation in some applications of key encapsulation.

3. A Definition of Key Encapsulation Scheme

3.1 Towards a Formal Definition

In the on-going development of theoretical support for cryptographic schemes, a formalism has been suggested for the hybrid cryptosystems, called KEM-DEM for Key Encapsulation *Mechanism* and Data Encapsulation Mechanism, see section 6.3 in reference [25], and references [26] and [27].

A key encapsulation *scheme* is a public key cryptography scheme that, given a public-key, derives a random symmetric key and provides a method of encrypting (encapsulating) and decrypting (decapsulating) that symmetric key. Our definition of key encapsulation borrows the notion of *label* from the definition of public key encryption in references [26] and [27]. This notion is augmented to fit the PEKE cryptosystem *first message*. Thus our definition looks like an abstract public key primitive that has a single instance. In spite of this, our definition is useful in conveying an abstraction of alleged PEKE security properties, allowing theoretical studies remote from implementation details but nonetheless addressing the unique aspects of PEKE.

-- start of definition --

A specification compliant to the present definition of a *key encapsulation scheme* shall include three algorithms:

- The probabilistic algorithm $Keygen$ is a key generation algorithm that takes a random seed as input and produces a key-pair $(pk; sk)$:
$$Keygen(seed_k)=(pk; sk)$$

- The probabilistic encapsulation algorithm, $Encaps$, takes the public-key pk, a random seed, and an arbitrarily valued label L as input and outputs a pair $(K; \psi)$, including ciphertext ψ,
$$Encaps(pk; seed_e; L)=(K; \psi)$$
such that
  - the symmetric key value K is dependent on the label L value in a non-malleable way, and
  - the symmetric key value K is also dependent on the public key pk value in a non-malleable way,

  these dependencies withstanding tampering with the pseudo-random component of the encapsulation algorithm.

- The deterministic decapsulation algorithm, $Decaps$, takes as input a ciphertext ψ, the label L, and the secret-key sk, and outputs a symmetric key value K' or the error symbol $\perp$.

$$Decaps(\psi; L; sk)=K' \text{ or } \perp$$

The compliant specification shall provide parameter size indications, i.e. a key size indication for the key generation algorithm and a label size indication. The symmetric key size should also be part of the key encapsulation specification.

Upon normal operation, $K'=K$, i.e. for a given pair $(pk; sk)$ output by the key generation algorithm, if L and ψ are such that $Encaps(pk; seed_e; L)=(K; \psi)$ for any $seed_e$ value, then $Decaps(\psi; L; sk)=K$.

The goal of key encapsulation security is to limit an attacker's ability to derive information about the key K from the public-key pk, the label L, and the ciphertext ψ.

-- end of definition --

3.2 Protocol and Usage Context

Protocol-wise, the label value origination flexibility provides an opportunity to ensure secret-key freshness for the originator of the label value. This is useful if the communications side that does the decapsulation algorithm also issues the label value.

We believe that this definition of key encapsulation contains desirable features for initial key establishment applications (for which we have hands-on experience with the SAKEM project), i.e.:

A) the client side of the application typically lacks a pre-existing trusted public key, so it implements the encapsulation computation, while the server side has a public key that is either widely distributed or backed by a security certificate signed by a trusted certification authority,

B) the client side can not arbitrarily choose the outcome of the key encapsulation transaction, key value K, which prevents some man-in-the-middle attacks,

C) a single primitive supports cases where key freshness is needed (i.e. label value is generated by the server) and store-and-forward connectivity (i.e. the label value is either fixed or selected by the client).

This usage context is different from session key establishment between peer parties both having a trusted public key, for which interesting theoretical advances are on-going, e.g. [32].

4. PEKE Specifications

4.1 Overview

The PEKE cryptosystem is a Rabin-Williams derivative (more specifically a Blum-Goldwasser probabilistic encryption [10] derivative), for key establishment applications. Accordingly, the PEKE security rests on the difficulty the of inverting the function $f(x)=x^2 \bmod N$, with N being the product of two distinct prime numbers, each congruent to 3 modulo 4. The application of this trapdoor function in a PEKE key encapsulation comprises an initialization of a BBS pseudo-random generator, the computation of a short sequence of BBS generator output, and an extra trapdoor function computation. The generator output is the outcome of the key encapsulation primitive, and the last trapdoor function result is the PEKE ciphertext. Details are given in the following sub-sections.

The probabilistic algorithm $Keygen$ is described in the section 4.2.1. The key encapsulation algorithm

$$Encaps(pk;\ seed_e;\ L)=(K;\ \psi)$$

is described in section 4.3 with the following adaptations to the notation

- the public key pk is N,
- the pseudo-random seed $seed_e$ initializes the random selection of $x_{B|}$
- the label L is $x_{A\to B}$,
- the output symmetric key K is w, and
- the ciphertext ψ is x_t,

so the encapsulation algorithm becomes

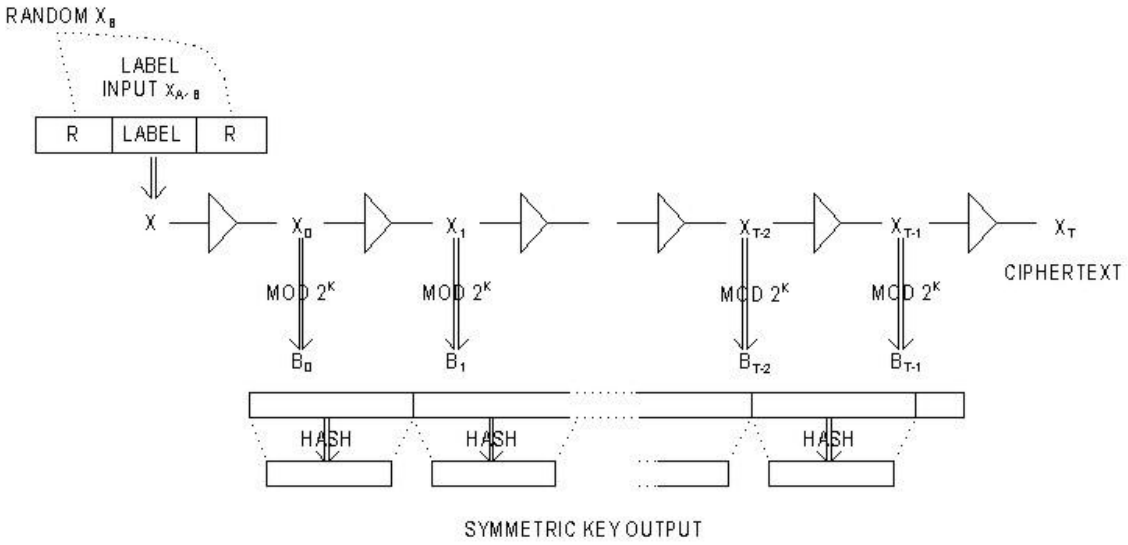
$$Encaps(N;\ x_{B|};\ x_{A\to B})=(w;\ x_t)$$

and the decapsulation algorithm becomes

$$Decaps(x_t;\ x_{A\to B};\ P,Q)=w \text{ or error indication } \perp$$

and is described in section 4.4.

In the present document, we retain the same notation as in [11], but with different explanations, and some parameter recommendations hinted by reference [3].



4.2 PEKE Parameters

4.2.1 Modulus Selection

The probabilistic algorithm *Keygen* is the key pair generation algorithm, much like the selection of an RSA modulus $N=P\times Q$ with P and Q being distinct prime numbers. An additional requirement specific to Rabin-Williams cryptosystems, including PEKE, is that the is that both P and Q must be congruent to 3 modulo 4. As usual, N is the public key and the factors P and Q are kept secret as the private key. The data format for the private key may be influenced by preprocessing on secret factors, see section 4.4.2. Contrary to the RSA cryptosystem, the PEKE public key has no public exponent component.

The mathematical analysis of the sequence generated by $x_{i+1}=x_i^2 \ mod \ N$ in reference [11] suggests additional number-theoretic conditions in order to produce long sequences. This issue should be ignored just as the RSA iterated encryption attack is not significant for selection

of RSA modulus. As is often the case with number-theoretic algorithms applied to huge integers, the simple approach to prime generation seems to avoid the number-theoretic pitfalls with an overwhelming probability. But see also reference [33] about the selection of large prime numbers for cryptographic applications.

4.2.2 Operational Parameters

In addition to the public key N, the PEKE cryptosystem requires some operational parameters, listed here with an indication of their use:

- key encapsulation label parameters: integers S and C, usage explained in section 4.3.1,
- BBS operating parameters: small integers k and t, usage explained in section 4.3.2, and
- an indication of which PEKE version applies (PEKEv0 or PEKEv1.25).

These operational parameters should be integrity-protected on the system where the key decapsulation occurs. This recommendation should be obvious from the need to protect the private key from logical implementation attacks.

4.3 PEKEv0 Encapsulation

4.3.1 BBS Seed Determination Using a Label Value

The PEKE label $x_{A\to B}$ is an integer in the range $0 \le x_{A\to B} < C$. The PEKE key encapsulation starts with the random selection of the local secret value $x_{B|}$ as an integer in the range $0 \le x_{B|} < \lfloor N/C \rfloor$. From these two values, a BBS generator initial value x is computed as

$$x = (\lfloor x_{B|}/S \rfloor \times S \times C) + (x_{A\to B} \times S) + (x_{B|} \bmod S)$$

This is a generic formula for a three-digits representation of x with the arbitrary bases $S \times C$ and S, the label $x_{A\to B}$ being the intermediate digit. The operational parameters S and C shall be such that $S \times C < N$. The parameters S and C may well be exact powers of two. The parameter S may have little, if any, impact of PEKE security, so $S=1$ seems reasonable.

In the Rabin-Williams encryption scheme described in reference [3], the *disclosed part* of the input to function $f(x)=x^2 \bmod N$, is less than half the size of N. We infer that the size of C should be less than half the size of N, i.e. $|C| < |N|/2$. The lower bound on C is a security parameter, reference [3] cites 128 bits as a reasonable size for a similar purpose s_0 field.

4.3.2 BBS Operation in PEKE Key Encapsulation

In a PEKE key encapsulation, the BBS pseudo-random sequence generator is first initialized as $x_0=x^2 \bmod N$ for a value x defined in the above subsection 4.3.1.

The operational parameters t and k define the bit length of the symmetric key output by PEKE key encapsulation. At each iteration of the BBS generator, the k lowest significant bits are retained in the symmetric key output. We suggest $k<|N|/3$ for PEKEv0 and $k<|N|/2$ for PEKEv1.25. A larger k allows a smaller t for a given symmetric key size $t \times k$, hence a more efficient key encapsulation algorithm.

Thus, a BBS generator sequence $x_0, x_1, x_2, \ldots, x_{t-1}$ is computed, where $x_i = x_{i-1}^2 \bmod N$ for $i>0$. The PEKE symmetric key output w is defined as $w=B_0|B_1|B_2|\ldots|B_{t-1}$, where $B_i = x_i \bmod 2^k$ and | represents concatenation of k-bit integers.

This concatenation of k-bit integers produces an octet string with the following logical transformation. The sequence $B_0|B_1|B_2|\ldots|B_{t-1}$ is read left to right as a single integer B expressed in the base 2^k:

$$B=B_0 \times 2^{(t-1)k}+B_1 \times 2^{(t-2)k}+\ldots+B_{t-2} \times 2^k+B_{t-1}$$

Since the symmetric key size $k \times t$ is not necessarily a multiple of 8 bits, from 1 to 7 bits of padding may be required at the end of symmetric key octet encoding. Let p denote this number of padding bits, then $p=0$ if $k \times t$ is a multiple of 8, otherwise $p=8-(k \times t \bmod 8)$. Finally the number $B \times 2^p$ is encoded as an octet string w with the most significant byte first convention. Note that the concatenation of k-bit integers was left undefined in reference [11].

The final computation $x_t = x_{t-1}^2 \bmod N$ gives the PEKE ciphertext x_t.

4.4 PEKEv0 Key Decapsulation

4.4.1 PEKE Key Decapsulation Principles

The PEKE decapsulation algorithm starts with the ciphertext x_t, and the operational parameter t to recover the four possible distinct modular square roots of x_0, i.e. computing the expression $x_t^{-2\wedge(t+1)} \bmod N$, giving four distinct values e, f, g, and h such that:

$$x_0 = e^2 \bmod N = f^2 \bmod N = g^2 \bmod N = h^2 \bmod N$$

This first step of the PEKE decapsulation is the mere application of the theorem 10b in reference [11], so implementation details provided in section 4.4.2 are informational.

One of the candidates e, f, g, and h may have been the x computed in a legitimate PEKE encapsulate operation using label $x_{A \to B}$. Each candidate is easily verified from the knowledge of $x_{A \to B}$, S, and C. If none of e, f, g, or h match this verification, the PEKE decapsulation outcome shall be the error signal.

The PEKE decapsulate algorithm proceeds with the recovery of PEKE symmetric key output w. This is done exactly as in the PEKE encapsulation algorithm in section 4.3.2, except that the $x_0 = x^2 \bmod N$ is replaced by $x_0 = e^2 \bmod N$.

In order to resist coarse timing attacks on the PEKE decapsulation algorithm, every computation step above should be performed, no matter the final outcome that may have been determined at an earlier step.

4.4.2 Key Decapsulation Implementation Details

This document subsection provides a step-by-step exposition of the mathematical programming needed to compute the four solutions to $x_t^{-2^{(t+1)}} \bmod N$ from x_t, t, and private factors P and Q.

For efficiency improvements, some pre-computation are recommended. Using the generalized Euclid algorithm, find integers a and b such that $a \times P + b \times Q = 1$. These numbers are used later in applying the Chinese remainder theorem. Then, compute the integers α and β:

$$\alpha = ((P+1)/4)^{(t+1)} \bmod (P-1)$$
$$\beta = ((Q+1)/4)^{(t+1)} \bmod (Q-1)$$

Then, a key decapsulation transaction starts with an x_t, and the following computations:

$$\mu = (x_t \bmod P)^{\alpha} \bmod P$$
$$\nu = (x_t \bmod Q)^{\beta} \bmod Q$$

The initiator may then compute the four solutions e, f, g, and h:

$$e = (b \times Q \times \mu + a \times P \times \nu) \bmod N$$
$$f = (b \times Q \times (P - \mu) + a \times P \times \nu) \bmod N$$
$$g = (b \times Q \times \mu + a \times P \times (Q - \nu)) \bmod N$$
$$h = (b \times Q \times (P - \mu) + a \times P \times (Q - \nu)) \bmod N$$

The programming of large number arithmetic and its validation requires special care in selection of proper optimization strategies. Although software libraries may provide ready to use programming tools in some context, some academic references may be cited for the interested

reader. Perhaps the single most important optimization in modular arithmetic is due to Peter L. Montgomery [34]; see also [35]. General textbooks on number theory covers topics such as the Euclid generalized algorithm, the Chinese remainder theorem and fast modular exponentiation. The general purpose division operation on large number is covered in [36].

4.5 PEKEv1.25 Specifications

The PEKEv1.25 specification fixes the original PEKE specification with an hash function applied to blocks of the symmetric key output. This hides the bits of the modular square root function. This was inspired by the use of a single hash function in the Rabin-Williams encryption scheme of [3]. The hash operation loses key bits, so that an adversary attempting to invert it would face a 2^{64}:1 ambiguity for PEKEv1.25.

The PEKEv1.25 rests on the original PEKE specification, with the an additional hash function applied to blocks of the symmetric key output to obtain the final PEKEv1.25 symmetric key output. For PEKEv1.25, the symmetric key size $k \times t$ shall be at least 320. To apply the hash function, the symmetric key output is segmented in blocks of 320 bits each, perhaps with 1 to 319 bits of symmetric key discarded at the end. Then, each block is processed as a message input to the SHA-256 hash algorithm ([37]), and each 256-bit output is retained. The final PEKEv1.25 symmetric key output is the concatenation of these 256-bit output blocks. The suffix "v1.25" in the name PEKEv1.25 comes from the 320/256=1.25 compression ratio in the hash operation.

Despite the current debate about the collision-resistance of the SHA-1 hash algorithm ([38], [39]), we see no reason to doubt about the use of SHA-256 in the PEKEv1.25 key encapsulation.

5. Conclusion

We presented the PEKE cryptosystem in its original form, and a modified version, inspired by recent security proofs for a Rabin-Williams encryption scheme. We state a few questions relevant to the topics covered in the present document:

Value of the key encapsulation scheme definition (above section 3.1).

- Q.1 Is the definition somehow useful in comparison with other definitions, e.g. the KEM definition in section 6.3 in reference [25]?
- Q.2 Is the definition formal enough?

Alternatives to PEKE for key encapsulation.

- Q.3 Are there any public key cryptography schemes compliant to the definition, besides PEKE?

Discrete logarithm key encapsulation schemes.

Q.4 Notably, is it possible to have a compliant scheme using a discrete log cryptosystem, preferably using an elliptic curve implementation?

Note that discrete log cryptosystems advantageously (over factoring-based schemes) use public domain parameters that are selected by means completely open to public scrutiny. Also elliptic curve efficiency is attractive and perhaps critical for long-term security.

Provable security for PEKEv1.25.

Q.5 Can PEKEv1.25 be supported by a fully developed computational complexity security proof?

6. References

[1] Moreau, Thierry, *Probabilistic Encryption Key Exchange*, Electronics Letters, Vol. 31, number 25 (7 December 1995), pp. 2166-2168. Available at http://www.connotech.com/PEKEELEC.HTM.

[2] See http://www.connotech.com/sakem_index.htm

[3] Dan Boneh, *Simplified OAEP for the RSA and Rabin Functions*, In Advances in Cryptology—Crypto 2001, 2001.

[4] Neal Koblitz, Alfred J. Menezes, *Another Look at "Provable Security"*, July 4, 2004

[5] Diffie, Bailey Whitfield, Hellman, Martin E., *New Directions in Cryptography*, IEEE Transactions in Information Theory, vol IT-22, 1976, pp 644-654

[6] Rivest, Ronald L., Shamir, Adi, Adleman, Leonard M., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, vol 21, 1978, pp 120-126

[7] Rabin, M.O., *Digital Signatures and Public Key Functions as Intractable as Factorization*, MIT Laboratory for computer science, TR 212, January 1979, pp 1-16

[8] Williams, Hugh C., *A Modification of RSA Public-Key Encryption*, IEEE Transactions on Information Theory, Vol IT-26, no. 6, November 1980, pp 726-729

[9] Goldwasser, Shafi, and Micali, Sylvio, *Probabilistic Encryption*, Journal of Computer and Systems Sciences, 28 (1984), pp 270-299

[10] Blum, M., and Goldwasser, S.: *An efficient probabilistic public-key encryption scheme*

which hides all partial information, Advances in Cryptology: Proc. of Crypto'84, 1985, (Springer-Verlag), pp. 289-299

[11] Blum, Leonore, Blum, Manuel, and Shub, M., *A Simple Unpredictable Pseudo-random Number Generator, SIAM Journal of Computing*, vol. 15, no. 2, May 1986, pp 364-383

[12] Vazirani, Umesh V., and Vazirani, Vijay V., *Efficient and Secure Pseudo-random Number Generation*, Proceedings of the 25th IEEE Symposium on the Foundations of Computer Science, 1984, pp 458-463

[13] ISO/IEC 9796:1991, *Information Technology - Security Techniques - Digital Signature Scheme Giving Message Recovery*

[14] Guillou, Louis Claude, Quisquater, Jean-Jacques, Walker, Mike, Landrock, Peter, and Shaer, Caroline, *Precautions taken against various potential attacks in ISO/IEC DIS 9796 'Digital signature scheme giving message recovery'*, Advances in Cryptology, Eurocrypt'90, Lecture Notes In Computer Science no. 473, pp 465-473

[15] Williams, Hugh C., *An M³ Public-Key Encryption Scheme*, Crypto'85, pp 358-368

[16] Lieberherr, Karl, *Uniform Complexity and Digital Signatures*, Theoretical Computer Science, Vol. 16 (1981), pp 99-110

[17] Williams, Hugh C., *Some Public-Key Crypto-Functions as Intractable as Factorization*, extended abstract, Crypto'84, pp 66-70

[18] Williams, Hugh C., *Some Public-Key Crypto-Functions as Intractable as Factorization*, Cryptologia, Vol. 9, number 3, July 1985, pp 223-368

[19] Chen, Kefei, *Authenticated encryption based on quadratic residue*, Electronics Letters, Vol. 34, No. 22, 1998

[20] Kurosawa, Kaoru, Ito, Toshiya, and Takeuchi, Masashi, *Public-Key Cryptosystem using a Reciprocal Number with the Same Intractability as Factoring a Large Number*, Cryptologia, Vol. 12, number 4, October 1988, pp 225-233

[21] Harn, Lein, and Kiesler, T., *Improved Rabin's Scheme with High Efficiency*, Electronics Letters, Vol. 25, no. 11, May 25th, 1989, pp 726-728 (see also erratum published in Electronics Letters, Vol. 25, no. 15, page 1016)

[22] Shimada, M., *Another Practical Public-Key Cryptosystem*, Electronic Letters, Vol. 28,

no. 23, November 5th, 1992, pp 2146

[23] Nyang, DaeHun, and Song, JooSeok, *Fast Digital Signature Scheme Based on the Quadratic Residute Problem*, Electronics Letters, 30th January 1997, Vol. 33, No. 3, pp205-206

[24] Victor Shoup, *Why Chosen Ciphertext Security Matters*, IBM Research division, Research Report, RZ 3076, 23/11/1998

[25] *NESSIE security report*, Version 2.0, February 19, 2003

[26] Victor Shoup, *A Proposal for an ISO Standard for Public Key Encryption (version 2.1)*, December 20, 2001

[27] Victor Shoup, editor, *ISO/IEC 18033-2, Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*, ISO/IEC JTC 1/SC 27/WG 2 Committee draft CD 18033-2, June 3, 2004

[28] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, *Recommendation for Key Management – Part 1: General*, NIST Special Publication 800-57, Draft, April, 2005

[29] Daniel J. Bernstein, *Proving Tight Security for Standard Rabin-Williams Signatures*, draft, 2003/09/26

[30] James Manger, *A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0*, in J. Kilian (Ed.): CRYPTO 2001, LNCS 2139, Springer-Verlag, 2001, pp. 230-238

[31] European Committee for Standardization (CEN), *Cryptographic module for CSP signing operations with backup - Protection profile - CMCSOB PP*, CWA 14167-2:2004, May 2004

[32] Hugo Krawczyky, *HMQV: A High-Performance Secure Diffe-Hellman Protocol*, IACR Cryptology ePrint Archive, June 13, 2005

[33] Maurer, Ueli M., *Fast Generation of Secure RSA-Moduli with Almost Maximal Diversity*, in Advances in Cryptology, Eurocrypt '92, Lecture Notes in Computer Science, no. 658, Springer-Verlag, 1992, pp 636-647

[34] Montgomery, Peter L., *Modular Multiplication Without Trial Division*, Mathemetics of

computations, Vol. 44, no. 170, April 1985, pp 519-522

[35] Dussé, Stephen R., and Kaliski, Burton S. Jr., *A Cryptographic Library for the Motorola DSP56000*, Advances in Cryptology, Eurocrypt'90, Lecture Notes In Computer Science no. 473, pp 230-244, Springer-Verlag, 1990

[36] Hansen, Per Brinch, *Multiple-length Division Revisited: a Tour of the Minefield*, Software Practice and Experience, vol. 24(6), June 1994, pp 579-601

[37] NIST, *Specifications for the Secure Hash Standard*, FIPS Publication 180-2 (+ Change Notice to include SHA-224), 2002 August 1

[38] NIST, *NIST Brief Comments on Recent Cryptanalytic Attacks on Secure Hashing Functions and the Continued Security Provided by SHA-1*, 2004/8/25

[39] Antoine Joux, *Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions*, in Advances in Cryptology - CRYPTO 2004, LNCS 3152, pp 306-316

[40] Housley, R., Ford, W., Polk, W., and Solo, D., *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, RFC 2459, January 1999

Annex A. PEKE Keys in X.509 Security Certificates

A.1 Background

According to RFC 2459 ([40]), an X.509 certificate "may convey a public key for any public key algorithm." In RFC 2459, section 7.3, a few well-known public key algorithms are specified. We specify herein the required interoperability provisions for using PEKE as a "subject public key algorithm" in X.509 security certificates. The present annex is thus provided mainly for the certification of PEKE public keys using X.509 security certificates. It may also be used independently of this purpose, e.g. as an encoding specifications for interchange of PEKE public keys.

Two relevant type definitions from RFC 2459 are reproduced below:

```
SubjectPublicKeyInfo  ::=  SEQUENCE  {
    algorithm           AlgorithmIdentifier,
    subjectPublicKey     BIT STRING  }

AlgorithmIdentifier  ::=  SEQUENCE  {
    algorithm             OBJECT IDENTIFIER,
    parameters             ANY DEFINED BY algorithm OPTIONAL  }
```

A.2 ASN.1 Encoding of PEKE Public Keys

The PEKE object identifier value for algorithm identification in X.509 certificate is defined herein as id-pekeID:

```
id-pekeID OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) 6 1 4 1 23742 1}
```

Where the id-pekeID algorithm identifier appears as the algorithm field in an RFC 2459 AlgorithmIdentifier defined type, the encoding shall either omit the parameters field (i.e. the AlgorithmIdentifier is a SEQUENCE of one component - the OBJECT IDENTIFIER id-pekeID) or include a NULL value as the parameters field.

The PEKE public key shall be encoded using the ASN.1 type PEKEPublicKey:

```
PEKEversion ::= ENUMERATED {
  pekeV0(0),  -- PEKEv0
  pekeV1-25(1) -- PEKEv1.25 -- }
```

```
PEKEpublicKey ::= SEQUENCE {
  peke-Modulus          INTEGER, -- N
  peke-Param-S          INTEGER, -- S
  peke-Param-C          INTEGER, -- C
  peke-Param-k          INTEGER, -- k
  peke-Param-t          INTEGER, -- t
  peke-Version            PEKEVersion  }
```

where the listed components are as defined in section 4.2.

A DER encoded PEKEpublicKey value fills the BIT STRING subjectPublicKey component in the RFC 2459 SubjectPublicKeyInfo defined type.

A.3 PEKE Public Key Usage Indications

If the keyUsage extension is present in an X.509 certificate which conveys a PEKE public key, the following values may be present: keyAgreement; encipherOnly; and decipherOnly. At most one of encipherOnly and decipherOnly shall be asserted in keyUsage extension (this provision is taken from section 7.3.2 of RFC 2459 where it applies to the Diffie-Hellman public key algorithm).