ESIGN 署名自己評価書

1　まえがき

本資料では、ディジタル署名を目的とする公開鍵暗号方式（暗号スキーム）「ESIGN 署名」の安全性および性能に関する自己評価について述べる。

なお、処理速度の評価においては、実装評価に加えて、特定のハードウェアやソフトウェア実装手法に依存しない最も客観的な評価方法である、基本的な演算（特定サイズの剰余乗算）の回数による机上評価を示す。

ESIGN 署名の長所は、非常に高い効率性を保持しており、かつ最強の安全性が証明されていることである。以下では、安全性と性能評価についてそれぞれ述べる。

2　安全性評価

2.1　概要

安全性については、ESIGN 署名は、RSA 仮定の近似版である e 乗根近似仮定 (AERP) とランダムオラクルモデルの下で、最強の意味で安全（適応的選択文書攻撃に対し存在的偽造不可）であることが証明できる。(安全性については次の節でもう少し詳しく述べることにする。)

安全性について、ESIGN 署名を他の代表的な安全性の証明のついた方式、RSA に基づく署名方式（例えば PSS や FDH-RSA），および楕円曲線に基づく署名方式 （例えば EC-Schnorr）、と比較した表を示す。

表 1: 安全性の比較

方式	安全性 （最強の意味で）	整数論的 仮定	ランダム関数 仮定
ESIGN	安全性証明つき	AERP	真にランダム
PSS or FDH-RSA	安全性証明つき	RSA	真にランダム
EC-Schnorr	安全性証明つき	楕円離散対数	真にランダム

2.2 理論的結果

この節では、ESIGN 署名の理論的な安全性評価を示す（なお、理論的な証明を行なった論文 [8] を本資料の最後に添付する。なお、この論文では、本資料の ESIGN 署名を TSH-ESIGN と称していることに注意）。

定義 2.1 $\mathcal{G}$ を *ESIGN* の鍵生成とする。e 乗根近似問題 *(AERP)* とは、$pk := \{n, e\} \leftarrow Gen(1^k)$ と $y \leftarrow_R \{0,1\}^{k-1}$ が与えられたとき、$0||y = [x^e \bmod n]^k$ となるような $x \in (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}$ を見つける問題である。

どのような確率的多項式時間アルゴリズム Adv に対しても、全ての定数 c, 十分に大きな値 k に対して

$$\Pr[Adv(k, n, e, y) \rightarrow x] < 1/k^c$$

が成立するとき、e 乗根近似問題は難しいという。ここで、$0||y = [x^e \bmod n]^k$ であり、確率は $\mathcal{G}$ と Adv の確率空間上で取られている。

e 乗根近似問題が難しいという仮定は、e 乗根近似仮定と呼ばれる。

定理 2.2 *ESIGN* は、e 乗根近似仮定が正しいという条件下で、ランダムオラクルモデルにおいて、適応的選択文書攻撃に対して存在的に偽造不可である。

補足 1:（e 乗根近似仮定について）独自の基本署名関数（暗号プリミティブ）である基本 ESIGN 署名関数を 15 年前に発表した [13, 12]。それ以来、この基本 ESIGN 署名関数の基本的安全性（一方向性）である e 乗根近似仮定について、様々な研究が行われてきたが、関数の次数 e が 4 以上の場合については、現在まで有効な攻撃は発見されておらず [5, 6, 9, 17]、提案者らは素因数分解以外に有効な攻撃法が無いと予想している。

補足 2:（$n = p^2q$ の素因数分解の困難性について）$n = p^2q$ の素因数分解が $n = pq$ よりも簡単かどうかについては分かってないが、$n = p^2q$ に特化されたいくつかのアルゴリズムが研究されている [14, 15, 16, 1]。しかし、これらのアルゴリズムはいずれも素因数分解の楕円曲線法でのアルゴリズムである。一方、$n = pq$ と $n = p^2q$ のいずれにおいても最高速の素因数分解アルゴリズムが数体ふるい法であり、このアルゴリズムの実行時間は n のサイズに依存して、素因数のサイズには依存しない。以上より、現時点では、$n = p^2q$ のサイズを $n = pq$ のサイズと同等にすれば、同等の安全性をもつものと考えられる。

3 実装評価

3.1 ハードウェアでの実装評価

- **使用したプロセス:**
  セルベース

- **設計環境:**
  Verilog-XL + DesignCompiler

- **リソース使用量:**
  約 25.6KG(@2NAND 面積換算) +メモリ (13312bit) 構成：[ランダムロジック+乗算器× 2 +加算器] + [メモリ (13312bit)]

- **速度評価:**
  クロック 30MHz での演算速度 (シミュレーションにより測定)

署名	9.8 ms
検証	2.5 ms

  ただし、鍵サイズ 1152 bit とする。

3.2 ソフトウェアでの実装評価

- **評価プラットホーム:**
  CPU: Pentium with MMX 266MHz
  OS: Trubo Linux version 4.0

- **記述言語:**
  C言語 (gcc version 2.91.60)
  多倍長整数演算ライブラリとして gnu mp (gmp version 3.0.1) を使用

- **メモリ使用量 (コード量):**

署名	48.589 Kbytes
検証	45.177 Kbytes

- **メモリ使用量 (ワークエリア):**

署名	512 Kbytes
検証	476 Kbytes

- **処理速度:**

署名	18.0 ms
検証	2.44 ms

ただし、鍵サイズ 1152 bit とする。

- **データサイズ:**

n のサイズ	1152 bits
e の値	1024
$hLen$	160 bits
$gLen$	160 bits
平文長	128 bits
公開鍵ファイルサイズ	329 bytes
秘密鍵ファイルサイズ	206 bytes
署名文ファイルサイズ	290 bytes

- **最適化の有無:**

コンパイル時の最適化は gcc -O3 を用いた。

本評価は、サンプルプログラム (call-6) として添付したものを実行した結果である。

mod n や mod p^2 の演算において、中国人の剰余定理を利用して高速に演算することが出来るが、今回の評価では、この高速化方法を用いていない。

したがって、チューンを行えば本評価より高速な実装が可能である。

4 性能の机上評価と他代表的方式との比較

ESIGN 署名の特徴は、その高効率にある。つまり、楕円曲線に基づく署名や RSA 署名方式など他の代表的な署名方式よりも効率良い。

次に、ESIGN の効率性を他の代表的な方式、RSA に基づく署名方式（例えば PSS or FDH-RSA）, および楕円曲線に基づく署名方式 （例えば EC-Schnorr と EC-DSA）、と比較する。それぞれの方式について、1152 ビット数の剰余乗算が必要な回数を評価することによって、鍵生成と検証に必要な処理量を見積もった。この比較においては、（拡張）バイナリ法や中国人剰余定理などの標準技術のみを考慮して評価を行なった。

パラメータは、ESIGN と RSA に基づく署名方式の法 n を 1152 ビット、楕円曲線に基づく署名方式の位数を 160 ビットと仮定した。また, RSA に基づく署名方式では $e = 2^{16}+1$, そして ESIGN では $e = 2^5$ を仮定した。なお、以下の表で、$\#M(1152)$ は 1152 bits の法の下での剰余乗算の回数を意味し、各処理量を $\#M(1152)$ に換算した値を示している。

表 2: 処理量の比較

方式	署名生成 ($\#M(1152)$)	署名検証 ($\#M(1152)$)
ESIGN	9	5
RSA に基づく方式	384	17
楕円曲線に基づく方式	41	48

参考文献

[1] Adleman, L.M. and McCurley, K.S.: Open Problems in Number Theoretic Complexity,II (open problems: C7, O7a and O7b), Proc. of ANTS-I, LNCS 877, Springer-Verlag, pp.291-322 (1995).

[2] Bellare, M. and Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, Proc. of the First ACM Conference on Computer and Communications Security, pp.62–73 (1993).

[3] Bellare, M. and Rogaway, P. : Optimal Asymmetric Encryption, Proc. of Eurocrypt'94, LNCS 950, Springer-Verlag pp.92-111 (1995).

[4] Bellare, M. and Rogaway, P.: The Exact Security of Digital Signatures – How to

Sign with RSA and Rabin, Proc. of Eurocrypt'96, LNCS 1070, Springer-Verlag, pp.399-416 (1996).

[5] Brickell, E. and DeLaurentis, J.: An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi, Proc. of Crypto'85, LNCS 218, Springer-Verlag, pp.28-32 (1986).

[6] Brickell, E. and Odlyzko: Cryptanalysis: A Survey of Recent Results, Chap.10, Contemporary Cryptology, Simmons (Ed.), IEEE Press, pp.501–540 (1991).

[7] Canetti, R., Goldreich, O. and Halevi, S.: The Random Oracle Methodology, Revisited, Proc. of STOC, ACM Press, pp.209–218 (1998).

[8] Fujisaki, E. and Okamoto, T.: Security of Efficient Digital Signature Scheme TSH-ESIGN, manuscript (1998 November).

[9] Girault, M., Toffin, P. and Vallée, B.: Computation of Approximate L-th Roots Modulo n and Application to Cryptography, Proc. of Crypto'88, LNCS 403, Springer-Verlag, pp.100-117 (1990).

[10] S. Goldwasser, S. Micali and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM J. on Computing, 17, pp.281–308, 1988.

[11] IEEE P1363 Draft, http://grouper.ieee.org/groups/1363/P1363/draft.html (1999).

[12] Okamoto, T.: A Fast Signature Scheme Based on Congruential Polynomial Operations, IEEE Trans. on Inform. Theory, IT-36, 1, pp.47-53 (1990).

[13] Okamoto, T. and Shiraishi, A.: A Fast Signature Scheme Based on Quadratic Inequalities, Proc. of the ACM Symposium on Security and Privacy, ACM Press (1985).

[14] Peralta, R.: Bleichenbacher's improvement for factoring numbers of the form $N = PQ^2$ (private communication) (1997).

[15] Peralta, R. and Okamoto, E.: Faster Factoring of Integers of a Special Form, IEICE Trans. Fundamentals, E79-A, 4, pp.489-493 (1996).

[16] Pollard, J.L.: Manuscript (1997).

[17] Vallée, B., Girault, M. and Toffin, P.: How to Guess L-th Roots Modulo n by Reducing Lattice Bases, Proc. of Conference of ISSAC-88 and AAECC-6 (1988).

Appendix

Security of Efficient Digital Signature Scheme TSH-ESIGN

Eiichiro Fujisaki Tatsuaki Okamoto

1 Introduction

In this manuscript, we will prove the security of the efficient digital signature scheme TSH-ESIGN.

2 Definitions

Definition 2.1 *Let A be a probabilistic algorithm and let $A(x_1, \ldots, x_n; r)$ be the result of A on input $(x_1, \ldots, x_n)$ and coins r. We define by $y \leftarrow A(x_1, \ldots, x_n)$ the experiment of picking r at random and letting y be $A(x_1, \ldots, x_n; r)$. If S is a finite set, let $y \leftarrow_R S$ be the operation of picking y at random and uniformly from finite set S. ε denotes the null symbol and, for list τ, $\tau \leftarrow \varepsilon$ denote the operation of letting list τ be empty. Moreover, $||$ denotes the concatenation operator, $|\cdot|$ denotes the size of bit length, i.e., $|y| := \lfloor \log_2 y \rfloor + 1$. For n-bit string x, $[x]^k$ and $[x]_k$ denote the most and least significant k bits of x, respectively ($k \leq n$).*

Definition 2.2 [Random Oracle Model] *We define by Ω the set of all maps from the set $\{0,1\}^*$ of finite strings to the set $\{0,1\}^\infty$ of infinite strings. Let H be a map from a set of an appropriate finite length (say : $\{0,1\}^a$) to a set of an appropriate finite length (say $\{0,1\}^b$). $H \leftarrow \Omega$ means that map H is chosen from Ω at random and uniformly, with restricting the domain to $\{0,1\}^a$ and the range to the first b bits of output.*

Definition 2.3 [Digital Signature Scheme] *A digital signature scheme is defined by a triple of algorithms (Gen, Sig, Ver) such that*

- *Key generation algorithm Gen is a probabilistic polynomial-time algorithm which on input 1^k ($k \in \mathbb{N}$) outputs a pair (pk, sk), called public key and secret key.*

- *Signing algorithm Sig is a probabilistic polynomial-time algorithm which on input $sk \leftarrow Gen(1^k)$ and a message $m \in \{0,1\}^*$ produces a string, $s \in \{0,1\}^*$, called the signature of m.*

- *Verification algorithm Ver is a probabilistic polynomial-time algorithm which on input $pk \leftarrow Gen(1^k)$ and a pair of a message and a signature, (m,s) returns 0 (i.e. invalid) or 1 (i.e. valid) to indicate whether or not the signature is valid. Here we insist that, for any (m,s) where $s \leftarrow Sig_{sk}(m)$, we require $Ver_{pk}(m,s) = 1$, otherwise $Ver_{pk}(m,s) = 0$ with overwhelming probability.*

Definition 2.4 [Forging Algorithm F] *Let $\Pi := (Gen, Sig, Ver)$ be a digital signature scheme. Let A be an algorithm which on input pk obtained by running the generator, Gen, has access to random hash function H and signing oracle Sig_{sk} and eventually outputs some string. We say that adversary A is a $(t, q_h, q_s, \epsilon(k))$-forger for $\Pi(1^k)$ if*

$$Adv_F^{\Pi}(k) := \Pr[H \leftarrow \Omega; (pk, sk) \leftarrow Gen(1^k) : F^{H, Sig_{sk}}(pk) \rightarrow (m,s)] \geq \epsilon(k),$$

where $Ver_{pk}(m,s) = 1$ and m is not included in the list of queries that F asks to $Sig_{sk}(\cdot)$, and, moreover, F completes within running time t, asking at most q_h queries to $H(\cdot)$ and at most q_s queries to $Sig_{sk}(\cdot)$.

3 TSH-ESIGN: ESIGN with Trisection Size Hash

This section introduces the digital signature scheme TSH-ESIGN.

3.1 Trisection Size Hash function h

Although it is necessary that the hash function used in our proposed signature scheme be ideal (or random oracle) to prove the security of the signature scheme, an arbitrary hash function (e.g., MD5 and SHA-1) is available for practical use. When the signature scheme is defined over $\boldsymbol{Z}/n\boldsymbol{Z}$, where $|n| = 3k$, the underlying hash function $h(\cdot)$ is defined as $h : \{0,1\}^* \rightarrow \{0,1\}^{k-1}$. (That is, the size of the image of hash h is around one third of the size of $|n|$.)

3.2 Key Generation algorithm Gen

Key generation algorithm Gen outputs, given security parameter 1^k ($k \in \boldsymbol{N}$), (pk, sk) where $pk = \{n, e\}$ and $sk = \{n, e, p, q\}$. The parameters, n, e, p and q, satisfy the following conditions:

- p, q ($p \neq q$) are prime numbers with k-bit length, i.e., $k = |p| = |q|$.
- $n := p^2 q$ with $3k$-bit length and $e > 4$.

3.3 Signature Generation $Sig_{sk}(m)$

The signature s of message m is computed by the signature algorithm $Sig_{sk}(\cdot)$ as follows:

Step 1 Pick r at random and uniformly from $(\boldsymbol{Z}/pq\boldsymbol{Z})\backslash p\boldsymbol{Z} := \{r \in \boldsymbol{Z}/pq\boldsymbol{Z} | \gcd(r, p) = 1\}$.

Step 2 Set $z \leftarrow (0||h(m)||0^{2k})$ and $\alpha \leftarrow (z - r^e) \bmod n$.

Step 3 Set (w_0, w_1) such that

$$w_0 = \lceil \frac{\alpha}{pq} \rceil, \tag{1}$$

$$w_1 = w_0 \cdot pq - \alpha. \tag{2}$$

If $w_1 \geq 2^{2k-1}$, then go back to **Step 1**.

Step 4 Set $t \leftarrow \frac{w_0}{e r^{e-1}} \bmod p$, and $s \leftarrow (r + tpq) \bmod n$.

Step 5 Output s.

3.4 Signature Verification $Ver_{pk}(m, s)$

For a pair of message m and signature s, the verification algorithm $Ver_{pk}(m, s)$ outputs *valid* (or 1) if

$$[s^e \bmod n]^{k+1} == 0||h(m)||0, \tag{3}$$

otherwise *invalid* (or 0).

Here we define by $Ver(h(m), pk)$ the set of *valid* signatures, namely, given $(h(m), pk)$, signatures that satisfy Equation (3).

Remark:

In practice, the verification equation can be replaced by

$$[s^e \bmod n]^k == 0||h(m). \tag{4}$$

The formal proof in the next section still works with minor modification of the approximate e-th root assumption.

4 Security

In this section, we examine the security of TSH-ESIGN. We first introduce a problem that is considered to be difficult to solve and then prove that breaking our scheme is as difficult as solving this problem.

We call the underlying problem the approximate e-th root problem (AERP). Roughly speaking, AERP is the problem, for given (y,e), of finding x such that $x^e \bmod n$ is approximately equivalent to y, i.e., $x^e \bmod n \approx y$.

Now let us define the approximate e-th root problem (AERP) and the breaking algorithm.

Definition 4.1 [Approximate e-th root problem (AERP)] *Let Gen be the generator of the TSH-ESIGN algorithm. Approximate e-th root problem (AERP) is, for given* $pk := \{n,e\} \leftarrow Gen(1^k)$ *and* $y \leftarrow_R \{0,1\}^{k-1}$, *to find* $x \in (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}$ *such that* $0||y||0 == [x^e \bmod n]^{k+1}$.

Definition 4.2 [AERP-Breaker B] *We say that adversary B is a* $(t,\epsilon(k))$*-AERP-breaker if*

$$Adv_B^{AERP}(k) := \Pr[(pk,sk) \leftarrow Gen(1^k); y \leftarrow_R \{0,1\}^{k-1} : B(pk,y) \rightarrow x] \geq \epsilon(k),$$

where $0||y||0 == [x^e \bmod n]^k$ *and B completes within running time t.*

The following theorem is the main result of this manuscript. This theorem implies that breaking AERP is as difficult as forging TSH-ESIGN (existentially forging under adaptive chosen message attacks).

Theorem 4.3 *If there exists a $(t(k), q_h(k), q_s(k), \epsilon(k))$-forger F for TSH-ESIGN, then there exists a $(t'(k), \epsilon'(k))$-AERP-Breaker B such that*

$$\begin{aligned} t'(k) &= t(k) + 4(q_h + q_s) \cdot \theta(k^3), \text{ and} \\ \epsilon'(k) &= (1/q_h) \cdot \epsilon(k). \end{aligned}$$

Proof:

Let $\Pi := (Gen, Sig, Ver)$ be a triple of the TSH-ESIGN algorithm and F be a (t, q_h, q_s, ϵ)-forging algorithm for TSH-ESIGN. We will present an AERP-Breaker B that includes F as an oracle. The aim of B is, given y, to find an approximate root x such that $0||y||0 == [x^e \bmod n]^{k+1}$. Recall that the advantage of ESIGN-breaker B is defined by

$$Adv_B^{\mathrm{AERP}}(k) := \Pr[(pk, sk) \leftarrow Gen(1^k); y \leftarrow_R \{0,1\}^{k-1} : B(pk, y) \rightarrow x].$$

Likewise recall that the advantage of forger F for TSH-ESIGN is defined by

$$Adv_F^{\Pi}(k) := \Pr[H \leftarrow \Omega; (pk, sk) \leftarrow Gen(1^k) : F^{H, Sig_{sk}}(pk) \rightarrow (m, s), \text{ where } s \in Ver(H(m), pk)],$$

where $Ver(H(m), pk)$ is defined as above in Sec.3.4 and m is not included in the list of queries that F asks to $Sig_{sk}(\cdot)$. Since forging algorithm F will make two kinds of queries, hash oracle queries and signing oracle queries, B must answer these queries by itself. Below we describe the specification of AERP-breaker B:

AERP-Breaker: $B(pk, y)$

set $\tau \leftarrow \varepsilon$ (empty) and $x \leftarrow \varepsilon$ (null);
set $l \leftarrow_R \{1, \ldots, q_h\}$, $i \leftarrow 0$ and $j \leftarrow 0$;
run $F(pk)$;
do while F does not ask query Q to $H(\cdot)$ nor ask to $Sig_{sk}(\cdot)$
if F asks query Q to $H(\cdot)$
$i++$; $j++$; (increment i, j);
if $j == l$
set $Q_l \leftarrow Q$;
put (Q_l, ε, y) in the list τ and answer F with y;
else if $Q \notin \tau$
set $Q_i \leftarrow Q$;

do while $0||*||0 == [x_i^e \bmod n]^{k+1}$,

$x_i \leftarrow_R (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}$;

set $(0||y_i||0) \leftarrow [x_i^e \bmod n]^{k+1}$;

put (Q_i, x_i, y_i) in the list τ and answer F with y_i;

else $(Q \in \tau)$

answer F with the corresponding $y' \in \tau$;

else if F asks query Q to $Sig_{sk}(\cdot)$

$i++$ (increment i);

if $Q_i = Q_l$

abort F and break;

else if $Q \notin \tau$

set $Q_i \leftarrow Q$;

do while $0||*||0 == [x_i^e \bmod n]^{k+1}$

$x_i \leftarrow_R (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}$;

set $(0||y_i||0) \leftarrow [x_i^e \bmod n]^{k+1}$;

put (Q_i, x_i, y_i) in the list τ and answer F with x_i;

else $(Q \in \tau)$

answer F with the corresponding $x' \in \tau$;

if F outputs (Q_l, x_l)

set $x \leftarrow x_l$;

return x

End.

Here '$*$' denotes an $(k-1)$-bit arbitrary string and τ denotes a list that records queries of F and the corresponding signatures and hash values that B makes. B starts with list τ empty and then records (Q_i, x_i, y_i) in τ by following the specification until aborting F or F ends asking queries.

Hereafter we explain the strategy of this specification, which is similar to that of FDH-RSA as presented by Bellare and Rogaway in [4].

First B picks up l uniformly from $\{1, \ldots, q_H\}$. B sets counters, i, j, as $i \leftarrow 0$ and $j \leftarrow 0$. Counter i is utilized for counting the total number of the queries that F asks to random oracle H and signing oracle Sig, while counter j is utilized for counting the number of F's asking to H.

- If F asks query Q to random oracle $H(\cdot)$, B increases i, j and then works as follows:
  - If the query is the l-th one, B sets $Q_l \leftarrow Q$, puts (Q_l, ε, y) in list τ, and answers F with y, where y is the instance input to B,
  - Else if the query has not been recorded in list τ, B sets $Q_i \leftarrow Q$, and then after executing **Experiment I**= [repeatedly picks up x_i from $(\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}$ until $0||*||0 == [x_i^e \bmod n]^{k+1}$, where '$*$' denotes an $(k-1)$-bit arbitrary string], makes an entry of (Q_i, x_i, y_i) in τ and answers F with y_i as the hash value of Q, where y_i is defined by $0||y_i||0 \leftarrow [x_i^e \bmod n]^{k+1}$.
  - Otherwise (i.e., Q is already in list τ), B answers F with the corresponding $y' \in \tau$.
- Similarly, if F asks query Q to signing oracle $Sig_{sk}(\cdot)$, B increases i and then works as follows:
  - If $Q_i = Q_l$, B aborts F and outputs ε (This means B failed to solve AERP),
  - Else if $Q \notin \tau$, B sets $Q_i \leftarrow Q$, and then after executing **Experiment I** makes an entry of (Q_i, x_i, y_i) in τ and answers F with x_i as the signing value of Q, where y_i is defined by $0||y_i||0 \leftarrow [x_i^e \bmod n]^{k+1}$.
  - Otherwise (i.e., Q is already in list τ), B answers F with the corresponding $x' \in \tau$.

Here we state that the distribution of (x_i, y_i) from **Experiment I** is identical to that of $(s, H(m))$ of signing algorithm $Sig_{sk}(\cdot)$ in the random oracle model. To prove this, we state the following two lemmas. To prove Lemma 4.5 is sufficient to prove this statement.

Lemma 4.4 *For given $h(m) \in \{0,1\}^{k-1}$ and $pk \leftarrow Gen(1^k)$, the following equation holds:*

$$\#Ver(h(m), pk) = \#\{r \in (\boldsymbol{Z}/pq\boldsymbol{Z})\backslash p\boldsymbol{Z} | 0 \leq (-\alpha \bmod pq) + pq < 2^{2k-1}\}, \qquad (5)$$

where $\alpha := ((0||h(m)||0^{2k}) - r^e) \bmod n$.

Sketch of Proof:

Recall that $Ver(h(m),pk) := \{s \in (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z} | [s^e \bmod n]^{k+1} == 0||h(m)||0\}$. Represent $s \in Ver(h(m),pk)$ by $s = r + tpq$ where $r \in (\boldsymbol{Z}/pq\boldsymbol{Z})\backslash p\boldsymbol{Z}$, and $t \in \boldsymbol{Z}/p\boldsymbol{Z}$. It is easy to check that (r,t) is the unique representation of s.

Let $Setr(h(m)) := \{r \in (\boldsymbol{Z}/pq\boldsymbol{Z})\backslash p\boldsymbol{Z} | 0 \leq (-\alpha \bmod pq) + pq < 2^{2k-1}\}$ where $\alpha := ((0||h(m)||0^{2k}) - r^e) \bmod n$. This makes it possible to make bijective map $Ver(h(m),pk) \rightarrow Setr(h(m))$ by $r := s \bmod pq$ and $t := (-\alpha \bmod pq) + pq$ where $\alpha := ((0||h(m)||0^{2k}) - r^e) \bmod n$. Therefore, $\#Ver(h(m),pk) = \#Setr(h(m))$. ¶

Lemma 4.5 *For given $s_0 \in Ver(h(m),pk)$, the following equation holds:*

$$\Pr[s \leftarrow Sig^h_{sk}(m) : s == s_0] = \Pr_{s \leftarrow_R (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}}[s == s_0 | s \in Ver(h(m),pk)] = \frac{1}{\#Ver(h(m),pk)}, \quad (6)$$

where $\Pr_{s \leftarrow_R (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z}}[s == s_0 | s \in Ver(h(m),pk)]$ denotes the conditional probability of Event $[s \leftarrow_R (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z} : s == s_0]$ given Event $[s \leftarrow_R (\boldsymbol{Z}/n\boldsymbol{Z})\backslash p\boldsymbol{Z} : s \in Ver(h(m),pk)]$.

Sketch of Proof:

¿From Lemma 4.4, the proof of this lemma is straightforward. ¶

¿From Lemma 4.5, we found that B can answer F by itself unless F asks Q_l as a signing query. Eventually, when F outputs (Q,s), Q is considered to be Q_j for some j. In the case of $j = l$, B succeeds to break AERP because $[s^e \bmod n]^{k+1} == y$, and the success probability $\epsilon'(k)$ is at least $(1/q_h) \cdot \epsilon(k)$. ¶