The PASSERINE Public Key Encryption and Authentication Mechanism

Markku-Juhani O. Saarinen

Aalto University
Department of Communications and Networking
P.O.Box 13000, 00076 Aalto, FINLAND
m.saarinen@tkk.fi

Abstract. PASSERINE [1] is a lightweight public key encryption mechanism which is based on a hybrid, randomized variant of the Rabin public key encryption scheme. Its design is targeted for extremely low-resource applications such as wireless sensor networks, RFID tags, embedded systems, and smart cards. As is the case with the Rabin scheme, the security of PASSERINE can be shown to be equivalent to factoring the public modulus. On many low-resource implementation platforms PASSERINE offers smaller transmission latency, hardware and software footprint and better encryption speed when compared to RSA or Elliptic Curve Cryptography. This is mainly due to the fact that PASSERINE implementations can avoid expensive big integer arithmetic in favor of a fully parallelizable CRT randomized-square operation. In order to reduce latency and memory requirements, PASSERINE uses Naccache-Shamir randomized multiplication, which is implemented with a system of simultaneous congruences modulo small coprime numbers. The PASSERINE private key operation is of comparable computational complexity to the RSA private key operation. The private key operation is typically performed by a computationally superior recipient such as a base station.

Keywords: Rabin Cryptosystem, Randomized Multiplication, RFID, Wireless Sensor Networks.

1 Introduction

Public key encryption is often viewed as unimplementable for extremely low-resource devices such as sensor network nodes and RFID tags. However, public key cryptography offers clear security advantages as fixed secret keys do not have to be shared between the two communicating parties. The PASSERINE public key encryption operation is very light, but the private key operation is approximately as computationally demanding as the private key operation of RSA.

For (RFID) authentication purposes a protocol can be devised that requires the tag to only perform public key encryption using the interrogator's public key.

In a military application a large number of sensors may be dispersed to an area of operations to lay passively dormant until an a particular combination of events triggers

[1] PASSERINE Version 0.5 September 23, 2010.

their activation. In such a scenario, key management with symmetric-only encryption may become exceedingly difficult. A single captured and reverse-engineered sensor unit may reveal all shared keys that it contains, possibly compromising the entire sensor network. Use of public-key cryptography simplifies key management and also reduces the need to protect keying information contained in the node. Each node only needs to store its unique identifier and the public key of the secure receiving station. The adversary can only impersonate a single physically captured sensor unit.

In this scenario the devices are controlled by a base station that stores their private identifiers. The devices only need to be able to perform the public key operation - to broadcast messages to the base station. A sensor unit can securely authenticate an another node with the aid of the trusted base station.

1.1 Previous Work

The use of Rabin encryption in low-resource platforms has been investigated by Shamir [17], Gaubatz et al. [6, 7] and more recently by Oren and Feldhofer [14]. The approaches considered in these papers differ significantly from PASSERINE; Gaubatz et al. do not consider randomized multiplication but only bit-serial multiplication. Shamir, Oren and Feldhofer use randomized multiplication but not CRT arithmetic nor payload encoding into the random mask. Systems described in [14, 17] require substantial amounts of real randomness, which may be difficult to generate in a resource-strained device. PASSERINE requires only a single random 128-bit key for each message. Naccache et al. [12] use randomized multiplication and CRT arithmetic (which they call Brugia-di Porto-Filipponi number system after [4]) in a low-resource implementation of a related identification protocol which was subsequently broken in [5].

2 The PASSERINE Randomized Rabin Cryptosystem

Rabin's public key cryptosystem [16] is in many ways similar to the RSA cryptosystem. Let n be a product of two large primes p and q. In order to facilitate implementation, these primes are often chosen so that $p \equiv q \equiv 3 \pmod 4$. To encrypt a message x, one simply squares it modulo the public modulus n:

$$z = x^2 \pmod n. \tag{1}$$

The Rabin private key operation requires computation of modular square roots and is of comparable complexity to the RSA private key algorithm. Since there are a total of four possible square roots ($\sqrt{z} \equiv \pm x \mod p$ and $\sqrt{z} \equiv \pm x \mod q$), a special mechanism is required in to mark and find the correct root. We refer to standard cryptography textbooks such as [9] for a discussion about implementation options.

The main distinguishing factor for the Rabin cryptosystem, in addition to being slightly faster than RSA in encryption, is that it is provably as secure as factoring. This equivalence may or may not hold for RSA [1, 3].

2.1 Shamir's Randomized Variant

In Eurocrypt '94 [17] Shamir proposed a randomized variant of the Rabin cryptosystem that avoids arithmetic $\bmod n$ by using a random masking variable $r > n$. The encryption operation is

$$z = x^2 + r \cdot n. \tag{2}$$

The private key operation is essentially the same as with the standard Rabin scheme.

Randomized multiplication was originally considered by Naccache [11], albeit for a different application. Shamir proved that this randomized variant has equivalent security properties to the standard version. The main drawback from avoiding modular arithmetic is that the ciphertext roughly doubles in size and that a large amount of high quality random bits must be generated for r. We avoid this problem using an encoding technique described in Section 2.3.

2.2 Arithmetic Modulo a Set of Coprime Numbers

A large majority of the implementation footprint of traditional public key encryption schemes such as RSA or ECC tends to be consumed by implementing large finite field multiplication and exponentiation. We avoid this by using arithmetic modulo a set of coprime numbers.

Let $b_1, b_2, \ldots, b_k$ denote a *base*, a set of coprime numbers, and $B = \prod_{i=1}^{k} b_i$ their product. The Chinese Remainder Theorem (CRT) states that any number x, $0 \leq x < B$ can be uniquely expressed as a vector x_i that represents a set of k congruences $x_i = x \mod b_i$ when $i = 1, 2, \ldots, k$. Furthermore, ring arithmetic modulo B can be performed in this domain. To compute the sum, difference or a product of two numbers $\bmod B$, all one needs to do is to is to add or multiply the individual vector components i, each $\bmod b_i$. Multiplication modulo B therefore has essentially linear complexity. Looking at Equation 2, one notices that when $z < B$, the entire public key computation can be performed in the CRT domain. This observation was first made in [4, 12].

Encryption Latency. One of the main advantages of a CRT implementation of PASSERINE is that serial transmission of encrypted data may be started immediately after the first word of $x^2 + r \cdot n$ has been computed. This is not the case with RSA or in ECC cryptography. This technique also helps to reduce the memory requirements of a PASSERINE implementation.

2.3 Carrying Payload Data in the Randomization Mask

An important and novel feature of PASSERINE is that r is used to carry payload data that has been encrypted using a random symmetric key, contained in x. This encoding technique allows us to essentially double the transmission bandwidth of the channel when compared to the original proposal by Shamir in [17].

3 Implementing PASSERINE Public Key Operation on a Low-Resource Platform

We targeted our implementation of PASSERINE encryption for low-end 8/16 - bit microprocessors and microcontrollers. We chose to use a 1025-bit public modulus, which offers a reasonable level of security. For highly sensitive data, a larger modulus should be used. For symmetric encryption, we use AES-128 in counter mode. The total code size was 1030 bytes when compiled for x86, including the relevant S-Box tables and the specially encoded prime table.

The CRT base (Section 2.2) was chosen to consist of 133 primes (6410th to 6542nd prime). These primes fit into sixteen-bit words as their numerical value ranges from 63929 to 65521. We chose to use primes rather than coprime numbers as there seems to be only a negligible encoding penalty from doing so. The encoding capacity is $\prod_{i=1}^{133} b_i \approx 2^{2125.70}$, which is only 2.30 bits short of the maximum channel capacity of $133 \times 16 = 2128$ bits.

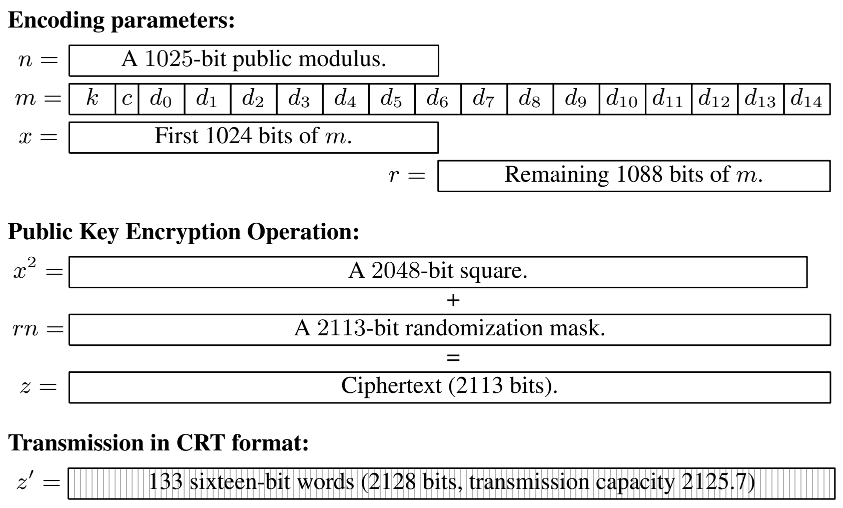

Fig. 1. Encoding of key and payload in PASSERINE. Encryption is actually performed using CRT representation (modulo small primes in base b), not in standard two's complement representation.

Figure 1 illustrates PASSERINE data encoding. We use AES-128 [13] in counter mode (CTR) for encryption of data blocks d_i

Table 1 describes the structure of the message m in our implementation. The first 1024 bits ($m[0..63]$) of the message are used as x and the latter 1088 bits ($m[64..131]$) as r in Equation 2. The 64-bit authenticator c should ideally be a secure MAC that is computed from all of the payload and key data. Our simple implementation sacrifices message integrity protection for implementation simplicity, and uses the result of a XOR operation between the upper and lower 64-bit halves of k as c. Adding good integrity protection with the EAX [2], CCM [15] or other authenticated mode is straightforward, but will slightly complicate implementation.

Table 1. Contents of the message m after symmetric encryption.

Symbol	**Words**	**Contents**
k	$m[0..7]$	A random 128-bit AES key, unique for each message.
c	$m[8..11]$	A 64-bit authenticator for message integrity and square root selection.
d_0	$m[12..19]$	First AES-encrypted block (payload bytes 0..15).
d_1	$m[20..27]$	Second AES-encrypted data block (payload bytes 16..31).
		...
d_{14}	$m[124..131]$	Fifteenth AES-encrypted data block (payload bytes 224..239).

4 PASSERINE Private Key Operation and Decryption

We implemented the private key operation in C using the OpenSSL library for both fast big number arithmetic and AES. The implementation required only about 230 code lines. In this section we will only give the relevant mathematics.

A straightforward method for converting the ciphertext to conventional two's complement binary representation is given in Equation 3. Here b_i is the base with $k = 133$ coprime numbers, $B = \prod_{i=1}^{k}$, and the CRT ciphertext vector z_i satisfies $0 \leq z_i < b_i$ for each i.

$$z = \left(\sum_{i=1}^{k} z_i \cdot \frac{B}{b_i} \cdot \left(\frac{B}{b_i} \right)_{b_i}^{-1} \right) \bmod B. \qquad (3)$$

The de-CRT coefficients $d_i = (B/b_i) \cdot (B/b_i)_{b_i}^{-1}$ in Equation 3 can be precomputed as they do not depend on the private parameters used.

Computing the square root. For decryption, one needs the private factorization pq of n. Rabin decryption is significantly easier to implement when $p \equiv q \equiv 3 \bmod 4$ and we will assume that this is the case. There are four square roots for every quadratic residue $\bmod\ pq$:

$$x_p = (z^{\frac{p+1}{4}} \bmod p) \cdot q \cdot q_p^{-1}. \qquad (4)$$

$$x_q = (z^{\frac{q+1}{4}} \bmod q) \cdot p \cdot p_q^{-1}. \qquad (5)$$

The four square roots of z are given by $x = \{x_p + x_q, x_p - x_q, -x_p + x_q, -x_p - x_q\} \pmod n$. The correct root can be recognized using the 64-bit authenticator c which is contained in x.

Symmetric decryption. Once the correct square root x is found, the mask r can be derived from

$$r = \frac{z - x^2}{n}. \tag{6}$$

We can then concatenate the two values and obtain the full message $m = x \mid\mid r$, which contains the symmetric decryption key and proceed to decrypt the entire data payload.

References

1. D. Aggarwal and U. Maurer. "Breaking RSA Generically Is Equivalent to Factoring." Eurocrypt 2009, LNCS 5479, Springer, pp. 36–53, 2009.
2. M. Bellare, P. Rogaway, and D. Wagner. "The EAX Mode of Operation." FSE 2004, LNCS 3017, Springer, pp. 389–407, 2004.
3. D. Boneh and R. Venkatesan. "Breaking RSA may not be equivalent to factoring." Eurocrypt '98, LNCS 1233, Springer, pp. 59–71, 1998.
4. O. Brugia, A. di Porto, and P. Filiponi. "Un metodo per migliorare l'efficienza degli algoritmi di generazione delle chiavi crittografiche basati sull'impiego di grandi numeri primi." Note Recesioni e Notizie, Ministero Poste e Telecommunicazioni, Vol. 33, No. 1-2, pp. 15–22, 1984.
5. J. Coron and D. Naccache. "Cryptanalysis of a Zero-Knowledge Identification Protocol of Eurocrypt '95." CT-RSA 2004, LNCS 2964, Springer, pp. 156–162, 2004.
6. G. Gaubatz, J. Kaps, E. Özturk, and B. Sunar. "State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks." PerCom 2005 Workshops, IEEE, pp. 146–150, 2005.
7. G. Gaubatz, J. Kaps, and B. Sunar. "Public key Cryptography in Sensor Networks – Revisited" Security in Ad-hoc and Sensor Networks – ESAS 2004, LNCS 3313, Springer, pp. 2–18, 2005.
8. T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D.A. Osvik, H. te Riele, A. Timofeev and P. Zimmermann. "Factorization of a 768-bit RSA modulus." `http://eprint.iacr.org/2010/006`. IACR Cryptology ePrint Archive: Report 2010/006, 2010.
9. A. Menezes, P. van Oorschot, and S. Vanstone. "Handbook of Applied Cryptography." CRC Press, 1996.
10. G. Lowe. "An Attack on the Needham-Schroeder Public-Key Authenticaion protocol." Information Processing Letters 56 (1995), Elsevier, pp. 131–131, 1995.
11. D. Naccache. "Method, Sender Apparatus And Receiver Apparatus For Modulo Operation." US patent: US5479511, 1995-12-26. European patent application: EP0611506, 1994-08-24. World publication: WO9309620, 1993.
12. D. Naccache, D. M'raihi, W. Wolfowicz, and A. di Porto. "Are Crypto-Accelerators Really Inevitable?" Eurocrypt '95, LNCS 921, Springer, pp. 404–409, 1995.
13. NIST. "Specification for the Advanced Encryption Standard (AES)" Federal Information Processing Standards Publication. FIPS-197, NIST, 2001.
14. Y. Oren and M. Feldhofer. "A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes." WiSec '09, ACM, pp. 59–68, 2009.
15. NIST. "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality." NIST Special Publication 800-38 C, NIST, 2004.
16. M. C. Rabin. "Digitalized Signatures and Public-Key Functions as Intractable as Factorization." MIT / LCS / TR-212, Massachusetts Institute of Technology, 1979.
17. A. Shamir. "Memory Efficient Variants of Public-Key Schemes for Smart Card Applications." Eurocrypt '94, LNCS 950, Springer, pp. 445–449, 1995.