Hidden Exponent RSA and Efficient Key Distribution

He Ge

Department of Computer Science and Engineering
University of North Texas, Denton, TX 76203
Email: ge@unt.edu

Abstract. In this paper we propose a variant of RSA public key scheme, called "Hidden Exponent RSA". Based on this new scheme, we devised an efficient key distribution/management scheme for secure communication among devices in the context of pervasive computing, with emphasis on the simplicity and efficiency of the protocol. We show the new scheme is secure under the strong RSA assumption.

Keywords: key management, key exchange, strong RSA assumption, pervasive computing

1 Introduction

In this paper we consider key distribution/management problem in the context of pervasive computing. Pervasive computing is an emerging trend in computation technology, i.e., computation can be carried out anywhere by any possible electronic devices collaboratively [11]. It includes areas such as mobile computing, distributed computing, etc. One example is wireless Ad Hoc network that could be deployed in both civil and military applications. In such network, wireless devices directly communicate with each other in peer-to-peer mode. For instance, in a sensor network, thousands of sensors are scattered in certain area to accomplish surveillance task. After deployment, all sensors are static and powered by batteries. Another instance is battalion team in military, where each members in the group carrying devices directly communicates with other members and the topology of the network is dynamically changing. Obviously, secure communication should be enforced in such networks due to military requirements.

The most direct solution is to preset common secret key(s) in these devices. However, one obvious disadvantage is that compromise of some devices will reveal all communication in network. Using public key cryptography, communication parties could generate session key for their communication. However, public key cryptography generally consumes lots resources such as computation and communication, which is not believed suitable for pervasive computing due to the limited capabilities of pervasive devices.

In this paper, we devise a lightweight key management scheme that might be appropriate for such situation. Even though based on public key cryptography, it is much more efficient than traditional key management scheme. The paper is organize as follows. In section 2 we propose a public key scheme that is a variant of well-known RSA. We introduce key management scheme in section 3. The performance analysis is given in section 4. We discuss the security properties in section 5. The paper concludes in section 6.

2 Hidden Exponent RSA Scheme

In 1978, Rivest, Shamir and Adlement proposed the well-known RSA public key scheme in the paper "A method for obtaining digital signature and public-key cryptosystems " [9]. This scheme elegantly solved the conjecture proposed by Diffie and Hellman about public key cryptography in their pioneer paper "New direction in cryptography" [5]. In this section, we first review RSA scheme. Then we introduce a new public key scheme that is a variant of RSA.

2.1 RSA Scheme

Alice picks two large random prime numbers p, q. She computes $n = pq$, and Euler's totient function $\phi(n) = (p-1)(q-1)$. She further picks a random $e < n$ that is relatively prime to $\phi(n)$, and computes d, the inverse of e modulo $\phi(n)$. Then (d, n) is the private key of Alice, and (e, n) is the public key. $(d, p, q, \phi(n))$ are kept secret by Alice. If another user Bob wants to send a message $m < n$ to Bob, he computes the ciphertext

$$c = m^e \pmod{n},$$

and sends it to Alice. Alice decrypts the ciphertext as

$$m' = c^d \pmod{n}.$$

If Alice and Bob follow the protocol,

$$m' = c^d = (m^e)^d = m^{ed} = m \pmod{n}.$$

In practice, p, q are set to 512-bit long. e is chosen quite small (3, or 65537 $(2^{16}+1)$). d is fairly large (above 1000 bit). Therefore, in RSA, encryption is efficient, while decryption is quite time-consuming.

2.2 Hidden Exponent RSA Scheme

We propose a variant of standard RSA scheme. Alice picks an element $g \in Z_n^*$ with large order. Z_n^* is the multiplicative group that contains all positive integer less than n and relatively prime to n. Therefore $\langle g \rangle$ is a cyclic subgroup of Z_n^*. Alice picks a random d which is relatively prime to the order of $\langle g \rangle$, and computes e which is the inverse of d modulo $|\langle g \rangle|$. Alice also computes $E = g^e \pmod{n}$. Now the public key is (E, g, n), and the private key is (d, n). In standard RSA, operations are over the group Z_n^*. In the new scheme, operations are over the subgroup of Z_n^*. This will not degrade the security of the scheme if g has sufficiently large order. However, e, public exponent in RSA, is being hidden as the discrete logarithm of E. That is why we call the new scheme "Hidden Exponent RSA".

The direct consequence of "hidden exponent" is that we can safely choose small decryption exponent d. In RSA, d has to be reasonably large to prevent small decryption exponent attack [10, 3]. When we hide e as the discrete logarithm of E, these attacks will not be effective anymore. However, we can immediately notice that encryption method does not work in the new scheme because e is not available now. Even though, Bob still can encrypt some random message to Alice. Suppose Bob picks a random r, computes $c = E^r \pmod{n}$ and sends it to Alice. Then Alice can decrypt it as $m' = c^d \pmod{n}$. If Bob and Alice follow the protocol, Alice will obtain

$$m' = c^d = (E^r)^d = (g^e)^{rd} = g^r \pmod{n}.$$

This shows Bob has sent a message $m = g^r \pmod{n}$ to Alice. However, Bob still cannot send meaningful message to Alice. We devise an encryption protocol which is similar to ElGamal encryption algorithm [6]. It works as follows.

- Using Alice's public key E, Bob encrypts a message $m < n$ as a pair (c_1, c_2) such that
$$c_1 = E^k \pmod{n}, \quad c_2 = Km \,(\mathrm{mod}\, n),$$
where $K = g^k \pmod{n}$ for a large random integer k.
- When Alice receives the ciphertext, she decrypts it as
$$K = c_1^d \,(\mathrm{mod}\, n), \quad m = c_2 K^{-1} \,(\mathrm{mod}\, n).$$

We have introduced the method to implement Hidden Exponent RSA scheme. The most advantage for the new scheme is the balance of computation overhead for encryption and decryption. Suppose we choose d, k 160 bit, the encryption needs about 320 modular multiplications, while decryption needs about 160 modular multiplications. The total costs are about 480 modular multiplications. In standard RSA scheme, the total computation overhead are above 1000 modular multiplications, most of which are on the decryption. The balance computation is useful in peer-to-peer communication such as sensor network. In such network, we would like computation overhead is evenly distributed among devices to extend whole life time of the system.

3 An Efficient Key Management Scheme

If we treat Alice as a key generation center (KGC), she then produces many such keypairs $(d_1, E_1), (d_2, E_2), \cdots, (d_k, E_k)$, and distributes each keypair to a device. Now each device has a hidden exponent RSA keypair. Thus, we obtain a key distribution scheme. However, some cautions should be taken at this moment.

When we apply a standard hidden exponent RSA to key distribution scheme, key forgery problem emerges. We should make sure that a valid keypair can only be generated by KGC. To do so, some restrictions need to be enforced. For instance, d must be a prime number. Otherwise, we can obtain multiple valid keypairs from one valid keypair. For example, suppose $d = d_1 d_2$, then $E = g^{e_1 e_2} \pmod{n}$. Obviously we can obtain two valid keypairs $(E_1 = g^{e_1} \pmod{n}, d_1)$, $(E_2 = g^{e_2} \pmod{n}, d_2)$.

In this section we introduce the method to extend hidden exponent RSA scheme to an efficient key distribution scheme.

3.1 Parameter Setting of KGC

We first review some definitions. A *Safe RSA Modulus* $n = pq$ is called safe if $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' also are prime numbers. An element $x \in Z_n^*$ is called a *quadratic residue* if there exists an $a \in Z_n^*$ such that $a^2 = x \pmod{n}$. The set of all quadratic residues of Z_n^* forms a cyclic subgroup of Z_n^*, which we denote by QR_n.

In the proposed key management scheme, a single key generation center (KGC) creates keypairs for devices. KGC's parameters are:

1. n, g: n is a safe RSA modulus such that $n = pq$, $p = 2p' + 1$, and $q = 2q' + 1$, where p and q are each 512 bit long, and p' and q' are prime. g is a generator of the cyclic group QR_n. n and g are public values while p and q are kept secret by KGC.
2. l_d, l_t: l_d is the length of decryption key which is 160 in our setting. l_t is the length of the tag which is used to uniquely identify a public key. We set $l_t = 24$. Tag t is set to prime in our scheme, therefore it can total represent 513,708 public keys which is large enough for our target application. For example, a sensor network at most has several thousands sensors.
3. A one-way hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^k$.

3.2 Keypairs Generation

KGC picks two random prime numbers d and t with lengths l_d and l_t, respectively, where t has not been previously used before. KGC computes

$$E = g^{et^{-1}} \pmod{n},$$

where e, t^{-1} are the inverses of d, t modulo $|QR_n| = p'q'$, respectively. d is the private key for the device, while (E, t) is the public key. t, called tag, is the unique identifier to represent a valid hidden exponent RSA keypair.

Through careful choice of parameters and some restriction, we have extended hidden exponent RSA scheme to a key distribution scheme. We will discuss the security in section 4. We can see, a nice feature is integrated into encryption scheme of hidden exponent RSA scheme: authentication. That is, when Bob sends a ciphertext encrypted by Alice's public key (E, t), he knows only authenticated Alice can decrypt this ciphertext. This is in contrast to ElGamal encryption scheme in which encryption does not provide authentication.

3.3 Authenticated Key Exchange

We introduce the most important application of hidden exponent RSA: session key generation. Suppose Alice and Bob hold their legitimate hidden exponent RSA keypairs. They implement an authenticated Diffie-Hellman key exchange [5] as follows.

1. Alice and Bob exchange their public keys: (E_a, t_a), and (E_b, t_b).
2. Alice checks t_b has length l_t, otherwise she aborts the protocol. Then she picks a random numbers r_a, calculates challenges $C_a = E_b^{t_b r_a} \pmod n$. Bob follows the same method to compute $C_b = E_a^{t_a r_b} \pmod n$. They exchange challenges.
3. Alice calculates $C_b^{d_a} \pmod n$ which is $g^{r_b} \pmod n$ if all parties follow the protocols. Then Alice can get $g^{r_a r_b} \pmod n$ which Bob is also supposed to obtain. Alice and Bob use this common value to derive a session key K.
4. Key confirmation step: Alice uses the session key to encrypt $\mathcal{H}(g^{r_b} \pmod n)$ using a symmetric encryption algorithm. Bob uses the session key to encrypt $\mathcal{H}(g^{r_a} \pmod n)$ using the same method. Alice and Bob exchange the confirmation message. If Alice/Bob find the decrypted value is equal to their own result, this finishes the key exchange protocol. Otherwise, the protocol is aborted.

In step 4, we use the hash value in the key confirmation to reduce the communication overhead. In a real application, the key confirmation step can be integrated into the subsequent data transmission. Therefore, only two rounds of message exchange is needed for the authenticated key exchange protocol.

3.4 Performance Analysis

We analyze the performance of key exchange protocol. We compare it with a solution based on traditional authenticated key exchange protocol [8, 4].

In a traditional scheme, each parties has its private key d_i, public key (e, n_i) in which e is almost the same for all parties. A key authentication center (KAC), with public key (e, n) and private key d, takes the responsibility to authenticate a user's public key n_i. KAC computes

$$h_i = (\mathcal{H}(n_i))^d \pmod n.$$

Then each party's authenticated key is in the format $(n_i||h_i)$. X.509 public key certificate is based on this technique [4].

When Alice and Bob wish to generate a session key, they first exchange their public key. Each side verifies that the received public key has been authenticated by KAC. After that, Alice picks a random integer r_a, and computes $m_a = g^{r_a} \pmod n$, and encrypts it as $C_a = m_a^e \pmod{n_b}$. Bob follows the same method to compute $C_b = (g^{r_b} \pmod n)^e \pmod{n_a}$. Alice and bob exchange the challenges and can finally compute the common value $g^{r_a r_b} \pmod n$, from which a session key can be derived.

We consider computation and communication overhead in the scheme. For simplicity, we let results of modulation have the length of modulus n which is 1024 bit in our setting. Thus, the message length in the protocol is 1024 bits. We also use the bit length of exponent to represent the number of modular multiplication. We choose 160 bit for random integer in the protocol. Based on these conditions, the new scheme needs 2048 bits communication payload for each party, while traditional scheme needs 3072 bits payload. The new scheme needs about 480 (160×3) modular multiplications for each party, while the traditional scheme needs about 1344 ($1024 + 160 \times 2$) modular multiplications.

This shows our key distribution scheme greatly improves the performance of the traditional scheme. This makes it more suitable for the application in pervasive computing due to the resource limitation for pervasive devices.

4 Security Properties of Proposed Scheme

We address the security of the scheme in this section. We mainly discuss key forgery issue. We first review a widely accepted and used security assumption, strong RSA assumption (for example, in [2, 7, 1]).

Assumption 1 (Strong RSA Assumption) *Let n be a RSA modulus. The* Strong RSA Assumption *says that for a random element $u \in Z_n^*$, no polynomial-time algorithm can find a pair (v, e) such that $e > 1$ and $v^e = u \pmod{n}$.*

Due to the strong RSA assumption, it is obvious that no one except KGC can create a a valid keypair (E, d, t) such that $E^{dt} = g \pmod{n}$.

However, we need address the issue of keypair forgery. In practice, more than one sensors could be compromised such that the keypairs are extracted. Since each valid keypair is uniquely identified by a tag t, We need to prove it is infeasible to forge a valid key with a new tag which is not created by KGC.

Theorem 1. *Under the strong RSA assumption, there exists no polynomial-time algorithm which takes a list of valid keypairs, $(d_1, E_1, t_1), (d_2, E_2, t_2), \ldots,$ (d_k, E_k, t_k) and produces a new keypair (d, E, t) such that $E^{dt} = g \pmod{n}$ and $t \neq t_i$ for $1 \leq i \leq k$.*

Proof. (Sketch) Suppose we have polynomial-time algorithm $\mathcal{A}$ which can compute a new valid keypair based on the available keypairs. Then given a random input (u, n), we can construct the following steps.

1. We pick random prime numbers $d_1, d_2, \ldots, d_k$ and $t_1, t_2, \ldots, t_k$ with the required bit lengths, and compute

$$r = d_1 d_2 ... d_k t_1 t_2 ... t_k,$$

$$g = u^r = u^{d_1 d_2 \ldots d_k t_1 t_2 \ldots t_k} \pmod{n}.$$

2. Next, we create k keys as follows:

$$\begin{aligned}
&(d_1,\ t_1,\ E_1 = u^{d_2 \ldots d_k t_2 \ldots t_k} \pmod n),\\
&(d_2,\ t_2,\ E_2 = u^{d_1 d_3 \ldots d_k t_1 t_3 \ldots t_k} \pmod n),\\
&\qquad\qquad \vdots\\
&(d_k,\ t_k,\ E_k = u^{d_1 d_2 \ldots d_{k-1} t_1 t_2 \ldots t_{k-1}} \pmod n)
\end{aligned}$$

Note that for all $i = 1, \ldots, k$, $E_i^{d_i t_i} = u^{d_1 d_2 \cdots d_k t_1 t_2 \cdots t_k} = u^r = g \pmod n$.

3. We use the algorithm $\mathcal{A}$ to create a new valid keypair (d, E, t) such that $E^{dt} = g \pmod n$. t will be different from all the t_i's, but will have the same length l_t. Therefore, t can not be an integer multiple of any of the t_i's, and since the t_i's are prime then it follows that $\mathrm{GCD}(t, t_1 t_2 \cdots t_k) = 1$. Furthermore, since the d_i's are prime and all longer than t, it follows that $\mathrm{GCD}(t, d_1 d_2 \cdots d_k) = 1$, and so $\mathrm{GCD}(t, r) = 1$. Therefore, we use the Extended GCD algorithm to find a and b such that

$$ar + bt = 1,$$

and let $y = (E^d)^a u^b$. Thus

$$y^t = (E^d)^{at} u^{bt} = u^{ar+bt} = u \pmod n,$$

Through these steps, assuming the existence of algorithm $\mathcal{A}$, we find a pair (y, t) such that $y^t = u \pmod n$ for a random u. However, this is infeasible under the strong RSA assumption. Therefore, we should not be able to find such algorithm $\mathcal{A}$ under the same assumption. This concludes the proof. □

As to the security of authenticated key exchange, we can identify the protocol includes two RSA challenge-response procedures which are secure under the standard RSA assumption.

5 Conclusion

In this paper, we proposed a public key scheme called "Hidden Exponent RSA". Based on this scheme, we devised a lightweight key management scheme which might be suitable for the application in pervasive computing. The new scheme has much lower communication and computation overhead compared to traditional key management scheme. Finally, we proved the new scheme is secure under the strong RSA assumption.

References

1. Ateniese, G., Camenisch, J., Joye, M., and Tsudik, G. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — Crypto* (2000), pp. 255–270.

2. Baric, N., and Pfitzmann, B. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — Eurocrypt* (1997), pp. 480–494.
3. Boneh, D., and Durfee, G. Cryptanalysis of RSA with private key d less than $n^{0.292}$. *IEEE Transactions on Information Theory 46*, 4 (2000), 1339–1349.
4. CCITT. Recommendation X.509: The directory-authentication framework, 1989. Blue book - Melbourne, Geneve.
5. Diffie, W., and Hellman, M. New direction in cryptography. *IEEE Transactions on Information Theory 11* (Nov. 1976), 644–654.
6. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — Crypto* (1984), pp. 10–18.
7. Fujisaki, E., and Okamoto, T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — Crypto* (1997), pp. 16–30.
8. Kohnfelder, L. Towards a practical public-key cryptosystem, May 1978. Bachelor's Thesis, M.I.T.
9. Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signature and public-key cryptosystems. In *Commnuications of the ACM* (Feb. 1978), vol. 21, pp. 120–126.
10. Wiener, M. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory 36*, 3 (1990), 553–558.
11. Wiener, M. Some computer science issues in ubiquitous computing. *Communications of the ACM 36*, 7 (1993), 75–84.