JOURNAL OF THEORETICAL PHYSICS & CRYPTOGRAPHY

www.IJTPC.org/Publications

A New Secret Sharing Based on finite automaton public key cryptosystem

A. Saeidi[a], M. M. Zahedi[b], A Nakhaei Amroodi[c]

[a]Department of Mathematics, Islamic Azad University, Kerman Branch, Kerman, Iran

[b]Department of Mathematics, Graduate University of Advanced Technology, Mahan-Kerman, Iran

[c]Department of Mathematics and Cryptography, Malek-Ashtar University of Technology, Isfahan, Iran

ABSTRACT

In this paper we consider the problem of secret sharing where the secret is encrypted using finite automaton public key cryptosystem (FAPKC3) and ciphertext is publicly available. In fact we present an (n,n)-threshold secret sharing scheme whose security can be reduced to the finite automaton public key cryptosystem. This is a strong property since the FAPKC3 is secure. The scheme has certain technical advantages: we did not use the discrete logarithm problem for verification whereas the most of cryptosystems use discrete logarithm problem for verification.

Keywords

Secret Sharing; Finite Automaton; Verification; Public Key Cryptosystem; Encryption; Decryption.

1. INTRODUCTION

Secret sharing schemes (SSS) are cryptographic primitives that allow a secret to be shared among a set of players such that only a subset with specified members can recover the secret. Secret sharing schemes were introduced by Shamir in 1979, with the scope of safeguarding encryption keys. SSS based on public key cryptosystems are ideal for sensitive systems such as missile launch codes, encryption keys, numbered bank accounts access control systems, e-voting, authentication protocols and etc (see [1], [2] and [3] and references there in). Since SSS is a noteworthy problem thus it has special importance in other fields such as graph theory [1] and etc. Recently Lein Harn has introduced an unconditional security of multi-secret sharing [5]. Also, Lein Harn and Miao Fuyoub have investigated Multilevel threshold secret sharing based on the Chinese Remainder Theorem in [4]. If in SSS, at least t players (up to n) with their shares can recover the secret then this SSS is called a (t,n)-threshold SSS. In other words a (t,n)-threshold secret sharing scheme requires the presence of at least t players (up to n) with their shares to recover the secret, while for any subset of t-1 or less players it is impossible to recover the secret. In this paper, we propose a threshold secret sharing scheme with t=n based on the finite automaton public key cryptosystem (FAPKC3). Tao Renji, Chen Shihua and Chen Xuemei introduced three public key cryptosystem based on finite automaton. They expressed a finite automaton public key cryptosystem [6] (FAPKC0) and two variants [7] (FAPKC1 and FAPKC2) based on invertibility theory of finite automata of which security rests on the difficulties of inversion of nonlinear finite automata. D. Dawei and et al. observed that under linear attacks FAPKC0 and FAPKC1 are insecure [8]. Also Bao and Igarashi found some weak inverses of finite automatons in [9]. In fact, nonlinear component has delay step > 0 in FAPKC2. Finally Tao Renji, Chen Shihua and Chen Xuemei proposed a new variant of FAPKC which is secure and still retains merits of the early FAPKC's such as the fast speed, relatively short public key, capability of digital signature, etc [10]. We investigate the (n,n)-threshold secret sharing scheme based on this FAPKC which is introduced in [10].

In the next section we explain finite automaton public key cryptosystem. Section 3 devotes to introducing a new verifiable secret sharing. Finally, we conclude our result with a brief discussion.

2. Preliminaries

In this section, we present the public key cryptosystem based on finite automaton. This cryptosystem is introduced in [10].

2.1 Finite Automata and Compound of them

In this paper, we denote the set of all words (finite sequences) over A including the empty word Λ by A^N, and the set of all infinite-length words (infinite sequences) over A by A^W for any set A.

Definition 1. A finite automaton M, is a quintuple $(X, Y, S, \Delta, \lambda\}$, where

- X (the input alphabet of M) is a nonempty finite set,
- Y (the output alphabet of M) is a nonempty finite set,
- S (the state alphabet of M) is a nonempty finite set,
- $\Delta: S\times X\to S$ (the next state function of M) is a single-valued function,
- and $\lambda: S\times X\to Y$ (the output function of M) is a single-valued function.

Sometimes M is called a sequential machine too. We can expand the domains of Δ and λ to $S\times X^N$ and $S\times\left(X^N \cup X^W\right)$, respectively, as follows.

$$\Delta(s,\Lambda)=s\ ,\ \Delta(s,\alpha x)=\Delta(\Delta(s,\alpha),x)$$

$$\lambda(s,\Lambda)=\Lambda\ ,\ \lambda(s,x\alpha')=\lambda(s,x)\lambda(\Delta(s,x),\alpha')$$

$$s\in S, x\in X, \alpha\in X^N, \alpha'\in X^N\cup X^W$$

Also a state sequence $s_0,s_1,..$, of M and an output sequence $y_0,y_1,..$, of M obtain using

$$s_{i+1}=\Delta(s_i,x_i)\ ,\ y_i=\lambda(s_i,x_i)\ ,\ i=0,1,2,...$$

Definition 2. A finite automaton $M = (X, Y, Y^k\times X^{h+1}, \Delta, \lambda)$ is said to be a (h, k)-order memory finite automaton, denoted by M_ϕ, if $y_i=\phi(y_{i-1},...,y_{i-k},x_i,...,x_{i-h})$ for i=0,1,2,... where ϕ is a mapping from $Y^k\times X^{h+1}$ to Y.

In other words in a (h, k)-order memory finite automaton M_ϕ:

$$\Delta((y_{i-1},...,y_{i-k},x_{i-1},...,x_{i-h}),x_i)=(y_i,...,y_{i-k+1},x_i,...,x_{i-h+1})$$

$$\lambda((y_{i-1},...,y_{i-k},x_{i-1},...,x_{i-h}),x_i)=y_i$$

$$y_i=\phi(y_{i-1},...,y_{i-k},x_i,x_{i-1},...,x_{i-h})$$

for i=0,1,2,.... If $k = 0$ then M_ϕ is called an h-order input memory finite automaton.

Definition 3. Supposed that g is a single-valued mapping from $U^r\times V^{p+1}\sigma_X^2$ to U, and f is a single-valued mapping from W^{t+1} to V.

$C'(M_f, M_g) = (W, U, U^r\times W^{p+t}, \Delta, \lambda)$ is called a (p+t, r)-order memory finite automaton if $u_i = g(u_{i-1},...,u_{i-r}, f(w_i,...,w_{i-t}),..., f(w_{-p},...,w_{i-p-t}))$ for i=0,1,2,.... In other words a (p+t, r)-order memory finite automaton $C'(M_f, M_g)$ is defined by

$$\Delta((u_{i-1},...,u_{i-r},w_{i-1},...,w_{i-p-t}),w)=(u_i,...,u_{i-r+1},w_i,...,w_{i-p-t+1})$$

$$\lambda((u_{i-1},...,u_{i-r},w_{i-1},...,w_{i-p-t}),w)=u_i$$

$$u_i=g(u_{i-1},...,u_{i-r},f(w_i,...,w_{i-t}),...,f(w_{-p},...,w_{i-p-t}))$$

$$w_i, w_{i-1},..., w_{i-p-t}\in W\ ,\ u_{-1},..., u_{i-r}\in U.$$

for i=0,1,2,.... Also we define

$$C'(M_0, M_1,..., M_n)=\begin{cases} C'(C'(M_0,M_1,...,M_{n-1}),M_n) & \text{for } n\geq 1\\ M_0 & \text{for } n=0\end{cases}$$

Remark. Obviously, the associative law holds but the commutative low does not hold.

Definition 4. Let $M=(X, Y, S, \Delta, \lambda\}$ and $M^*=(Y, X, S^*, \Delta^*, \lambda^*\}$ be two (h, k)-order and $(\tau + k, h)$-order memory finite automaton respectively. We call M and M^* satisfy the first automatons inevrtibility conditions (AIC1) if the following conditions hold

(a) For any state $s_i = (y_{i-1},..., y_{i-k}, x_{i-1},..., x_{i-h})$ of M and any x_i, x_{i+1}, ... in X, if $y_iy_{i+1}... = \lambda(s_i, x_ix_{i+1}...)$, then $\lambda^*(s_i^*, y_{i+\tau}y_{i+\tau+1}...) = x_ix_{i+1}...$, where $s_i^*=(x_{i-1},...,x_{i-h},y_{i+\tau-1},...,y_{i-k})$.

(b) For any state $s_i^{**}=(x'_{i-1},...,x'_{i-h},y_{i-1},...,y_{i-\tau-k})$ in $\left\{\Delta^*=(s^*,y_i,...,y_{i-\tau-1})\middle| s^*\in S^*, y_i,...,y_{i-\tau-1}\in Y\right\}$ and state $s_i=(y_{i-1-\tau},...,y_{i-k-\tau},x'_{i-1},...,x'_{i-h})$ of M, for any $y_i, y_{i+1},...$ in Y, if $x'_ix'_{i+1}...=\lambda^*(s_i^{**}, y_iy_{i+1}...)$, then $\lambda(s_i, x'_ix'_{i+1}...)=y_{i-\tau}...y_{i-1}y_iy_{i+1}...$.

Definition 5. Let $M = (X, X, S, \Delta, \lambda)$ and $M^* = (X, X, S^*, \Delta^*, \lambda^*)$ be two h-order input and (τ, h)-order memory finite automaton respectively. We call M and M^* satisfy the second automatons inevrtibility conditions (AIC2) if the following conditions hold

(a) For any state $s_{i+1}=(x_{i-1},...,x_{i-h})$ of M and any x_i, x_{i+1}, ... in X, if $x'_ix'_{i+1}...=\lambda(s_{i+1}, x_ix_{i+1}...)$ then $\lambda^*(s_i^*, x'_{i+\tau}x'_{i+\tau+1}...)=x_ix_{i+1}...$ where $s_i^*=(x_{i-1},...,x_{i-h},x'_{i+\tau-1},...,x'_i)$.

(b) For any state $s^*=(x_{i-1},...,x_{i-h},x'_{i-1},...,x'_{i-\tau})$ of M^* and for any $x'_i, x'_{i+1},...$ in X, if $x_ix_{i+1}...=\lambda^*(s^*, x'_ix'_{i+1}...)$, then $\lambda((x_{i+\tau-1},...,x_{i+\tau-h}),x_{i+\tau}x_{i+\tau+1}...)=x'_ix'_{i+1}...$.

2.2 Basic Algorithm

Suppose that all users use the same alphabet to communicate each other. A user, say *Bob*, chooses his own public key and secret key as follows. he

1. Constructs a (h_0, k_0)-order memory finite automaton $M = (X, Y, S_0, \Delta_0, \lambda_0)$ and a $(\tau_0 + k_0, h_0)$-order memory finite automaton $M_0^* = (Y, X, S_0^*, \Delta_0^*, \lambda_0^*)$ satisfying in AIC1.
2. Constructs an h_1-order input memory finite automaton $M_1 = (X, X, S_1, \Delta_1, \lambda_1)$ and a (τ_1, h_1)-order memory finite automaton $M_1^* = (X, X, S_1^*, \Delta_1^*, \lambda_1^*)$ satisfying in AIC2.
3. Constructs the finite automaton $C'(M_1, M_0) = (X, Y, S, \Delta, \lambda)$ from M_1 and M_0.
4. Computes $x'_{-h_0,e}, \ldots, x'_{-1,e} = \lambda_1((x_{-h_0-1,e}, \ldots, x_{-h_0-h_1,e}), x_{-h_0,e}, \ldots, x_{-1,e})$ where $s_e = (y_{-1,e}, \ldots, y_{-k_0,e}, x_{-1,e}, \ldots, x_{-h_0-h_1,e})$ is an arbitrary state of $C'(M_1, M_0)$.
5. Puts $s_{0,d}^{out} = (x'_{-1,e}, \ldots, x'_{-h_0,e}), s_{1,d}^{out} = (x_{-1,e}, \ldots, x_{-h_1,e})$.
6. Sets the public key $= \{C'(M_1, M_0), s_e, \tau_0 + \tau_1\}$ and the private key $= \{M_1^*, M_0^*, s_{1,d}^{out}, s_{0,d}^{out}, \tau_0, \tau_1\}$.

Encryption: Any user, say *Alice*, wants to send the user *Bob* a plaintext $x_0, \ldots, x_n$. Using Bob's public key, *Alice*

- Suffixes any $\tau_0 + \tau_1$ digits, say $x_{n+1} \cdots x_{n+\tau_0+\tau_1}$, to the plaintext $x_0, \ldots, x_n$.
- Computes the ciphertext $y_0 \cdots y_{n+\tau_0+\tau_1}$ as follows: $y_0 \cdots y_{n+\tau_0+\tau_1} = \lambda(s_e, x_0 \cdots x_{n+\tau_0+\tau_1})$.

Decryption: To retrieve the plaintext from the ciphertext $y_0 \cdots y_{n+\tau_0+\tau_1}$, Using Bob's private key, *Bob*

- Computes $x'_0, \ldots, x'_{n+\tau_1} = \lambda_0^*((x'_{-1,e}, \ldots, x'_{-h_0,e}, y_{\tau_0-1}, \ldots y_0, y_{-1,e}, \ldots, y_{-k_0,e}), y_{\tau_0} \ldots y_{n+\tau_0+\tau_1})$.
- Retrieves the plaintext $x_0, \ldots, x_n = \lambda_1^*((x_{-1,e}, \ldots, x_{-h_1,e}, x'_{\tau_1-1}, \ldots, x'_0), x'_{\tau_1}, \ldots, x'_{n+\tau_1})$

Theorem 1. The above cryptosystem works correctly.

For proof of theorem 1 see [10]. Theorem 1 assures us that the process of encryption and decryption are inverse of each other. Since this cryptosystem is secure and its implementation is easy [10], thus we use it as the base of secret sharing scheme.

3. Verifiable Secret Sharing

3.1 Initialization and Construction phase

We show the players of the scheme with P_i, $0 \le i \le n-1$ and the dealer of the scheme with D. The dealer should compute the shares of the secret and distribute them to all the players in the scheme. Our scheme is a (n,n)-threshold scheme, so in order to recover the secret, the *n* players have to participate with their shares. Let the secret be $x_0, x_1, \ldots, x_n$. The dealer should share it between the *n* players, the dealer executes in the following manner: he

I) Constructs a (h_i, k_i)-order memory finite automaton $M_i = (X, Y, S_i, \Delta_i, \lambda_i)$ and a $(\tau_i + k_i, h_i)$-order memory finite automaton $M_i^* = (Y, X, S_i^*, \Delta_i^*, \lambda_i^*)$, for $i=2k$, $k=0,1,\ldots,n-1$, satisfying AIC1.

II) Constructs a h_i-order memory finite automaton $M_i = (X, Y, S_i, \Delta_i, \lambda_i)$ and a (τ_i, h_i)-order memory finite automaton $M_i^* = (Y, X, S_i^*, \Delta_i^*, \lambda_i^*)$, for $i=2k+1$, $k=0,1,\ldots,n-1$, satisfying AIC2.

III) Construct the finite automaton $C'(M_{2i-1}, M_{2i-2}) = (X, Y, S, \Delta, \lambda)$ from M_{2i-1} and M_{2i-2} for $i=1,\ldots,n$.

IV) Choose arbitrary $s_e = (y_{-1,e}, \ldots, y_{-k_{i,e}}, x_{-1,e}, \ldots, x_{-h_{i-1}-h_i,e})$ of $C'(M_{2i-1}, M_{2i-2})$ for $i=1,\ldots,n$. Compute

$$x'_{-h_i,e}, \ldots, x'_{-1,e} = \lambda_i\left((x_{-h_i-1,e}, \ldots, x_{-h_i-h_i,e}), x_{-h_i,e}, \ldots, x_{-1,e}\right)$$

and put

$$s_{2i-1,d}^{out} = \left(x'_{-1,e}, \ldots, x'_{-h_i,e}\right), s_{2i-2,d}^{out} = \left(x_{-1,e}, \ldots, x_{-h_i,e}\right)$$

V) Computes and publishes the ciphertext *S* from following procedure:

S:=secret;

for *i*=1 to *n* do

- Add $\tau_{2i-1} + \tau_{2i}$ digits, say $x_{n+1} \cdots x_{n+\tau_{2i-1}+\tau_{2i}}$, to the *S*.
- Compute the ciphertext $y_0 \cdots y_{n+\tau_{2i-1}+\tau_{2i}}$ as follows:

$y_0 \cdots y_{n+\tau_{2i-1}+\tau_{2i}} = \lambda(s_e, x_0 \cdots x_{n+\tau_{2i-1}+\tau_{2i}})$

where λ is the output function of $C'(M_{2i-1}, M_{2i-2})$.

- S:= $y_0 \cdots y_{n+\tau_{2i-1}+\tau_{2i}}$.

end do.

VI) Distributes i (the priority), M_{2i-1}^{*}, M_{2i-2}^{*}, $s_{2i-1,d}^{out}$, $s_{2i-2,d}^{out}$, τ_{2i-1} and τ_{2i-2} to P_i for $i=1,\ldots,n$.

3.2 Verification and recovery phase

To recover the secret, all the shares need. In fact, only n players can recover the secret. Any P_i has two automaton M_{2i-1}^{*} and M_{2i-2}^{*} for $i=1,\ldots,n$. We call the input of M_{2i-1}^{*} as the input of P_i and the output of M_{2i-2}^{*} as the output of P_i. The recovery with verification process is as follows:

i) Sort all the players by their priority.

ii) Each P_i for i=1,…,n can verify whether he is valid or not by checking the following condition:

Input-Output Equality Condition (IOEC): The input of P_j and the output of P_i should not equal for $j=i,\ldots,n$ (see proposition 1).

Proposition 1. The complexity of IOEC is $O(n^2)$.

Proof.

Any P_i should check the IOEC i times, for $i=n,\ldots,2$. Also P_1 should check the IOEC n-1 times. Therefore, we need $1+2+\ldots+(n-2)+(n-1)+(n-1)=\frac{n(n+1)}{2}-1$ times to check IOEC.

The above proposition gives us the number of searches in step ii).

iii) If all the shares are valid, then they can be used to recover the secret $x_0,x_1,\ldots,x_n$ by the decryption of $S=y_0,\ldots,y_{n+T}$ where $T=\sum_{i=0}^{2n-1}\tau_i$ as follows:

for $i=n$ down to 1 do

S:=the decryption of S by P_i .

end do.

The correctness of above encryption/decryption is due to the following Theorem:

Theorem 2. The decryption of $S=y_0,\ldots,y_{n+T}$ is $x_0,x_1,\ldots,x_n$ where S is the ciphertext of the secret $x_0,x_1,\ldots,x_n$ as mentioned in subsection 3.1.

Proof.

It is sufficient to apply n times theorem 1.

4. ACKNOWLEDGMENTS

This research was financed from the budget of Islamic Azad University Kerman Branch-Iran in the form of research design of “Post Quantum cryptography based on lattices”. Therefore, the authors are highly grateful to the Department of Mathematics, Islamic Azad University, Kerman Branch, Kerman, Iran for providing an excellent research environment in which to conduct this research.

5. Conclusions and Future works

We constructed a simple, efficient unanimous consent SSS based on the famous Finite automaton public key cryptosystem. We showed that the scheme does not allow recovering the secret if at least one player is missing and this situation reduces to the break automaton public key cryptosystem, which is believed to be secure. The scheme offers the possibility for the players to check if all the shares distributed by the dealer are valid without using the discrete logarithm problem which is usually used in SSS. There is still a lot of work to be done in order to improve the capabilities of the scheme: it would be good to find a (k,n) variant of the scheme with k≠n and a way to make it multisecret (to allow sharing several secrets instead of one secret shared on each round).

6. REFERENCES

[1] Salomaa A. Public-Key Cryptography, Springer-Verlag, Berlin, 1990.

[2] Chum C. S., Zhang X., 2013, Hash function-based secret sharing scheme designs, Security and Communication Networks; 6:584–592.

[3] Ou D. and Sun W., 2014, Reversible AMBTC-based secret sharing scheme with abilities of two decryptions, Journal of Visual Communication and Image Representation, 25(5): 1222-1239.

[4] Harna L. and Fuyoub M., 2014, Multilevel threshold secret sharing based on the Chinese Remainder Theorem, Information Processing Letters, 114(9): 504-509.

[5] Harn L., 2014, Secure secret reconstruction and multi-secret sharing schemes with unconditional security, Security and Communication Networks, 7:567–573.

[6] Renji T., Shihua C., 1985, A finite automaton public key cryptosystem and digital signatures, Chinese Journal of Computers (in Chinese), 8(6), 401-409.

[7] Renji T., Shihua C., 1986, Two varieties of finite automaton public key cryptosystem and digital signatures. Journal of Computer Science and Technology, 1(1), 9-18.

[8] Dawei D., Kui W., and Huanguo Z. , 1995, Cryptanalysis on a finite automaton public key cryptosystem, Science in China, Series A (Chinese Edition), 25(11), 1226-1232.

[9] Feng B., and Igarashi Y., 1995, Break finite automata public key cryptosystem, Automata, Languages and Programming, 147-158.

[10] Renji T., Shihua C. and Xuemei C., 1997, FAPKC3: A New Finite Automaton Public Key Cryptosystem, Journal of Computer Science and Technology, 12(4), 289-305.