REPORT CPKI - DESIGN AND IMPLEMENTATION

20.12.2023

—

PAPER/SUBJECT NAME: INFORMATION SECURITY

SEMESTER - 1

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF DELHI

SUBMITTED BY:

AMBEDKAR (11), ARYAN (15), HARSH (20),

NACHIKET (35), PRADEEP (39), ROHAN (47)

Overview

The purpose of this document is to present a detailed description of the papers assigned and analyzed under the umbrella concept of Central Public Key Infrastructure, and its design and implementation. Following are paper-wise introductions and analysis of various papers undertaken in our Information Security project.

META - PKI

(NACHIKET)

This paper proposes an algorithm for verification of public key certificates in a decentralized public key infrastructure – Meta-PKI.

Reference: Simplifying Dynamic Public Key Certificate Graph for Certification Path Building in Distributed Public Key Infrastructure

Shohei Kakei, Yoshiaki Shiraishi, Shoichi Saito

Motivation

A mesh PKI can eliminate a single point of failure because the responsibility of a single CA is distributed across multiple CAs.

However, this is likely to complicate certification path building to determine the order in which certificates are to be verified.

The paper addresses this issue and gives an algorithm to simplify certification path building.

Advantages

In the case of building certification paths with the naive method (Yen's algorithm), the processing time grew exponentially, while the increase of the processing time is kept constant in the proposed method.

The proposed algorithm can compute the entire certification paths more efficiently that the naive method.

Drawbacks / Shortcomings

1. In the proposed algorithm, the authors suggest a threshold for graph partitioning. However, no further discussion is made on the optimal choice of the said threshold.
2. Due to constraints and policies for cross-certification, some public key certificates may be considered invalid even if these certificates are in certification paths.
   To make certification path building flexible, it is conceivable that all certification paths are presented to a certificate verifier, and the verifier chooses whether or not to accept the certification paths by the constraints and the policies.
   However, the processing load on PKI clients will be high, and the management of the constraints and the policies of the PKI clients can be complicated.

PKI-4-IOT

(NACHIKET, ROHAN)

This paper proposes an automated certificate enrolment protocol for use with IoT devices – PKI4IoT – public key infrastructure for internet of things.

Reference: PKI4IoT: Towards public key infrastructure for the Internet of Things

Joel Höglunda, Samuel Lindemer , Martin Furuhedb, Shahid Razaa

Motivation

1. Security risk: The IoT presents great security risks owing to the scale in which it is implemented. Lack of authentication mechanisms in embedded devices, on such a scale, may lead to weaker protection.
2. Resource constraint: We cannot make a direct switch from other authentication methods to PKI since modern PKI is not designed for constrained environments.

Advantages

1. Communication overhead: PKI4IoT reduces the size of X.509 certificates by more than 50% using profiling and CBOR encoding. It also reduces the number of certificates

needed for trust establishment by allowing the client to indicate that it only expects the server's own certificate.

2. Energy consumption: PKI4IoT reduces the energy expenditure for the client node by up to 58% compared to the case where a full certificate chain is sent. This is achieved by combining certificate compression and shortening techniques. The savings are more significant in multi-hop networks with lossy links.
3. Round trip times: PKI4IoT marginally improves the expected handshake time for the client node by reducing the number of bytes transmitted. The improvement is more noticeable in noisy radio environments, where packet losses and retransmissions are more frequent.

Drawbacks / Shortcomings

1. A lot of alternative methods to solve the problem of implementing PKI in embedded systems have been overlooked. The reasons for exclusions are mostly assertions; no strict arguments have been made for exclusion of alternative security solutions.
2. The authors claim PKI to be the only security solution that ensures maximum security in constrained devices/ or any device for that matter.
   (a) PKI still faces threat from quantum computers, capable of breaking classical crypto algorithms. This has been acknowledged by the authors, while also stating that such an issue is not a concern for the near future.
   (b) Other alternative security solutions exist, which could be implemented instead. Hardware security modules, multi-factor authentication, blockchain etc. to name a few. While these may require additional components in the constrained devices, even PKI4IoT requires comparable additional components.

AN EFFICIENT KEY MANAGEMENT AND MULTI-LAYERED SECURITY FRAMEWORK FOR SCADA SYSTEMS

(AMBEDKAR)

SCADA stands for Supervisory Control and Data Acquisition. It refers to a system of software and hardware elements that allows organizations to control industrial processes, monitor and gather data in real-time from remote locations, and interact with devices such as valves, pumps, sensors, and more, typically found in industrial environments like manufacturing plants, power generation facilities, water treatment plants, and infrastructure systems.

Shohei Kakei, Yoshiaki Shiraishi, Shoichi Saito

Key Elements of the Framework

1. Key Generation: Session keys are created using a combination of random numbers, current date & time, and fractions of prime numbers. These elements are hashed to generate session keys, and their distribution is secured using MACSALT.
2. Key Distribution: The exchange of session keys, encrypted random numbers, and MACSALT is facilitated securely over the communication channel using asymmetric key cryptography.
3. Key Extraction: At the receiver end, the private and public keys are used to validate authentication and confidentiality, extracting the necessary elements to generate session keys

Secure Information Exchange

• Control messages exchanged between devices are kept short and encrypted using a Vernam cipher, requiring distinct keys for each message.

• Different methods for symmetric key generation:

1. Hybrid Multi-Layered Architecture: Utilizes both symmetric and asymmetric key cryptography for secure communication.
2. Random Prime Number Generator: Generates session keys based on random primes for encryption and decryption.
3. Prime Counter: Utilizes a prime counter to enhance execution speed in generating session keys.
4. Hash Chaining: Provides high security and availability by using pre-shared keys and hash functions for key generation.

Conclusion

The proposed model intends to elevate security across various industrial sectors like water plants, power stations, chemical industries, and transportation systems. Successful implementation of this framework promises enhanced remote monitoring and control capabilities while fortifying the entire system against potential breaches.

A DIGITAL SIGNATURE BASED ON PKI TO AUTHENTICATE AND SECURE EXCHANGING DATA USED IN WATER BOREHOLES INTELLIGENT DECISION SUPPORT SYSTEM

(HARSH)

The authors introduce a digital signature system based on Public Key Infrastructure (PKI). The system is designed to secure the exchange of critical data within the DSS (decision support system), ensuring the reliability and integrity of the information shared among the users.

Reference: K. Semar-Bitah, F. Z. Bouderbala and Y. Ghebghoub, "A Digital Signature Based on PKI To Authentication and Secure Exchanging data used in water boreholes intelligent Decision Support System," 2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS), BLIDA, Algeria, 2023, pp. 1-5, doi: 10.1109/ICAECCS56710.2023.10105057.

Motivation

Water scarcity is a critical issue, particularly in regions like the Northern Sahara, where groundwater is a lifeline for drinking, irrigation, and industrial purposes. To achieve better exploitation, the authors suggest implementing a Decision Support System (DSS) to sustain the management of water boreholes in the southern part of the country. The data exchanged in their system is confidential and must be secured.

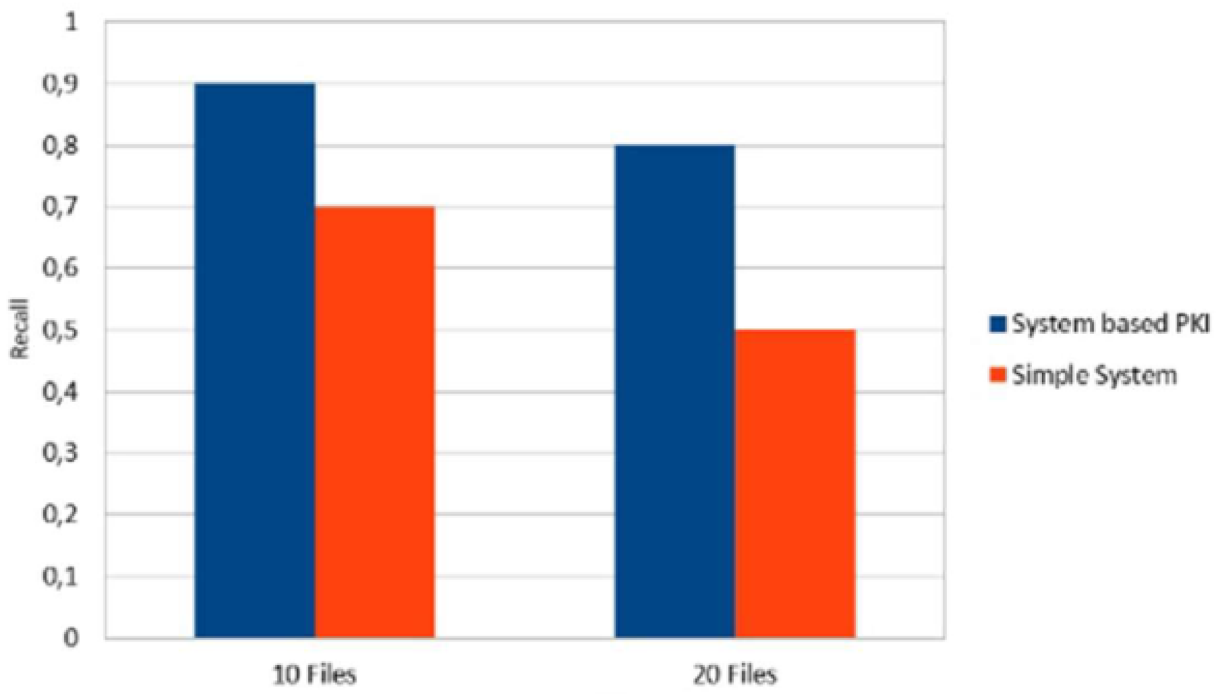
Implementation

- Language used: JAVA. IDE: NetBeans
- Linux server was configured using open SSL to serve as the backbone of their Public Key Infrastructure.
- SHA-256 used as the hash function to have the message digest of the data on our document.
- RSA used for encryption/decryption with the private key/ public key.
- iText5, an open-source library was used to manipulate and create PDF documents.

Actors and their roles

Actor	Role
Administrator	Manages users and their certificates.
Signatory	Signs administrative documents using his private key.
Receiver or requestor	Signature requester. Verifies the signature when receiving the signed document.

Results



For evaluation purposes, recall measure was used.

$$Recall = \frac{NumberOfAuthorizedAccessSignedKPI}{FilesNumberOfSignedPKIFiles}$$

The Digital Signature Standard

(HARSH)

The research work was proposed by NIST (National Institute of Standards and Technology) aiming at building an algorithm which can be used to make digital signatures, instead of traditional written signatures. It can be used to verify both the identity and authenticity of the originator. The name of the Algorithm was Digital Signature Algorithm (DSA). DSA comprised two parts :

1. Signature Generation: by using a private key.
2. Signature Verification: by using a public key.

Implementation

DSA Parameters:

p = a prime modulus where $2^{511} < 2^{512}$

q = a prime divisor of p - 1 , where $2^{159} < q < 2^{160}$.

g = $h^{(p-1)/q}$ mod p, where h is any integer with $0 < h < p$ such that $h^{(p-1)/q}$ mod $p > 1$.

Private Key → **x** = an integer with $0 < x < q$.

Public Key → **y** = g^x mod p.

m = the message to be signed and transmitted.

k = a random integer with $0 < k < q$.

H = a one-way hash function.

The integers **p**, **q** and **g** can be public and can be common to a group of users. **x** and **k** must be secret. **k** must be changed for each signature.

Signature Generation & Verification:

To generate and sign the message **m** user chooses a random integer k and computes:

r = $(g^k \bmod p) \bmod q$

s = $(k^{-1}(H(m) + xr)) \bmod q$

The values **r** and **s** constitute the signature of the message. These are transmitted along with the message **m** to the recipient.

m', **r'**, and **s'** be the **received versions** of **m**, **r**, and **s**, respectively, and let **y** be the **public key** of the sender.

To verify the signature, the recipient first checks to see that **$0 < r' < q$** and **$0 < s' < q$; if either condition is violated the signature is rejected**. If these two conditions are satisfied, the recipient computes

$\boldsymbol{w} = (s')^{-1} \bmod q$

$\boldsymbol{u1} = ((H(m'))w) \bmod q$

$\boldsymbol{u2} = ((r')w) \bmod q$

$\boldsymbol{v} = (((g)^{u1}(y)^{u2}) \bmod p) \bmod q.$

If v = r', the signature is verified

If v does not equal r', the signature is not valid.

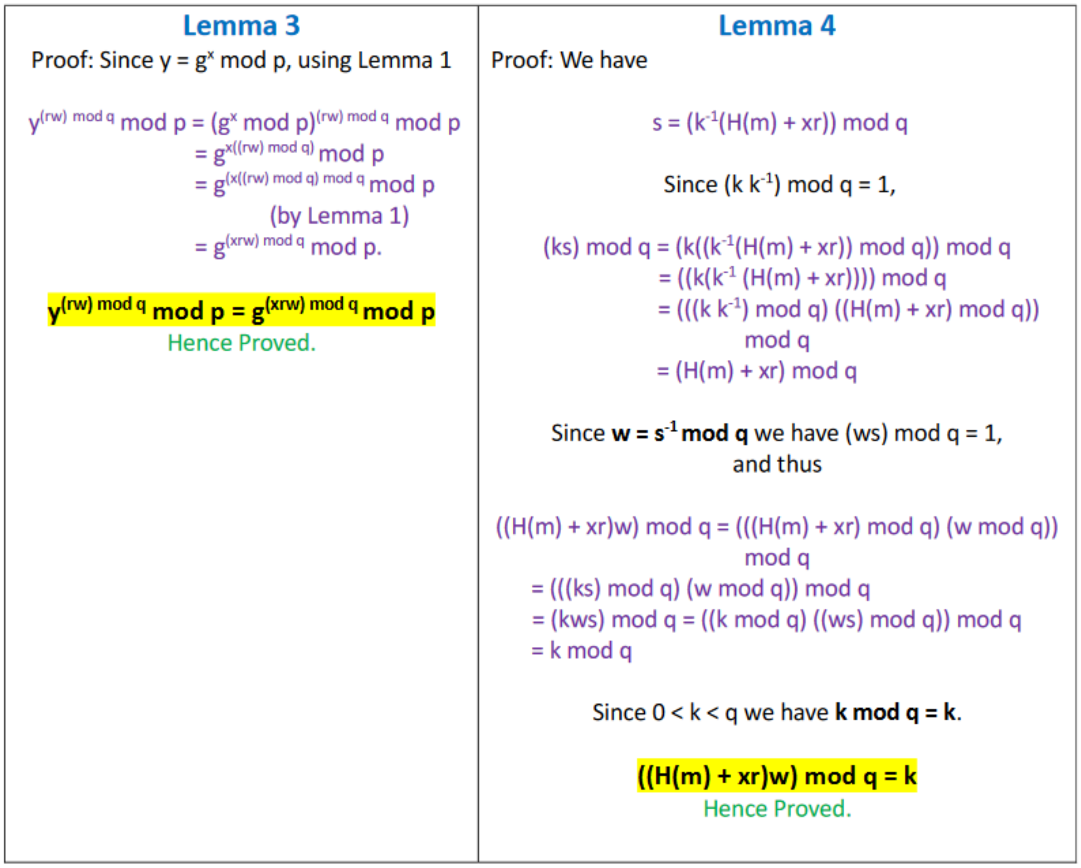
Proof v = r'

The purpose is to prove that we get v = r', when m'=m, r'=r, s'=s.

This is proved by **proving 4 Lemmas :**

Lemma 1 $g^{t} \bmod p = g^{t \bmod q} \bmod p$	**Lemma 2** $g^{a \bmod q + b \bmod q} \bmod p = g^{(a+b) \bmod q} \bmod p$
Lemma 3 $y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p$	**Lemma 4** $((H(m) + xr)w) \bmod q = k$

Lemma 1	**Lemma 2**
Proof: By the Euler/Fermat theorem, since h is relatively prime to p, we have $h^{p-1} \bmod p = 1$. Hence for any nonnegative integer n, $g^{nq} \bmod p = (h^{(p-1)/q} \bmod p)^{nq} \bmod p$ $= h^{((p-1)/q)nq} \bmod p$ $= ((h^{(p-1)} \bmod p)^{n})$ $= 1^{n} \bmod p$ $= 1.$ Thus, for any nonnegative integers **n** and **z,** $g^{nq+z} \bmod p = (g^{nq}\, g^{z}) \bmod p$ $= ((g^{nq} \bmod p)\,(g^{z} \bmod p)) \bmod p$ $= g^{z} \bmod p.$ Any nonnegative integer **t** can be represented uniquely as **t = nq + z** where **n** and **z** are nonnegative integers. Then $g^{t} \bmod p = g^{t \bmod q} \bmod p$. Also, z = t mod q. Hence Proved.	Proof: By Lemma 1 we have $g^{a \bmod q + b \bmod q} \bmod p = g^{(a \bmod q + b \bmod q) \bmod q} \bmod p$ $= g^{(a+b) \bmod q} \bmod p$ $g^{a \bmod q + b \bmod q} \bmod p = g^{(a+b) \bmod q} \bmod p$ Hence Proved.



Lemma 3

Proof: Since $y = g^x \bmod p$, using Lemma 1

$$
\begin{aligned}
y^{(rw) \bmod q} \bmod p &= (g^x \bmod p)^{(rw) \bmod q} \bmod p \\
&= g^{x((rw) \bmod q)} \bmod p \\
&= g^{(x((rw) \bmod q) \bmod q} \bmod p \\
&\qquad \text{(by Lemma 1)} \\
&= g^{(xrw) \bmod q} \bmod p.
\end{aligned}
$$

$$\mathbf{y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p}$$

Hence Proved.

Lemma 4

Proof: We have

$$s = (k^{-1}(H(m) + xr)) \bmod q$$

Since $(k\,k^{-1}) \bmod q = 1$,

$$
\begin{aligned}
(ks) \bmod q &= (k((k^{-1}(H(m) + xr)) \bmod q)) \bmod q \\
&= ((k(k^{-1} (H(m) + xr)))) \bmod q \\
&= (((k\,k^{-1}) \bmod q)\ ((H(m) + xr) \bmod q)) \\
&\quad \bmod q \\
&= (H(m) + xr) \bmod q
\end{aligned}
$$

Since $\mathbf{w = s^{-1} \bmod q}$ we have $(ws) \bmod q = 1$, and thus

$$
\begin{aligned}
((H(m) + xr)w) \bmod q &= (((H(m) + xr) \bmod q)\ (w \bmod q)) \\
&\quad \bmod q \\
&= (((ks) \bmod q)\ (w \bmod q)) \bmod q \\
&= (kws) \bmod q = ((k \bmod q)\ ((ws) \bmod q)) \bmod q \\
&= k \bmod q
\end{aligned}
$$

Since $0 < k < q$ we have $\mathbf{k \bmod q = k}$.

$$\mathbf{((H(m) + xr)w) \bmod q = k}$$

Hence Proved.

Conclusion

This Algorithm can be used at the e-platforms where data interchanges between authorities, and where the authentication of its generating source and assurance of its integrity, is at utmost priority. For e.g. E-mails, electronic data interchange, etc. For Hardware implementations, the reference given is of "VLSI Implementation of Public Key Encryption Algorithms" by Orton, Roy, Scott, Peppard and Tavares from the Proceedings of CRYPTO=86. For Software implementations, the reference given is of “A Cryptographic Library for the Motorola DSP56000" by Dusse and Kaliski from the Proceedings of EUROCRYPT-90 and "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor" by Barrett from the Proceedings of CRYPTO-86.

PKI-Based Cryptography for Secure Cloud Data Storage Using ECC

(ARYAN)

Data safety in cloud storage and focus on the risks of unauthorized access and data tampering.

To solve the security issues, the paper looks at a few suggested solutions.

- using a type of encryption called attribute-based encryption
- electronic IDs for authentication
- a method called user hierarchy encryption
- a specific kind of cryptography called elliptic curve cryptography (ECC)

Certificate Authority (CA) and PKI Enabled Application (PEA)

The Certification Authority (CA) is an important part of a system called Public Key Infrastructure (PKI). It's like a trusted agent that creates and gives out certificates.

When people or organizations want certificates, they share their details with the CA. The CA checks and confirms these details through a secure process.

Methodology

Data in the cloud is divided into private and shared parts

- private part is for storing sensitive data used only by the user
- shared part allows data to be shared with multiple authenticated users

Three cloud data storage models are designed based on this concept:

- Secure Data Backup model
- Secure Data Sharing One to One
- Secure Data Sharing One to Many.

Key generation function DKGen(S, Data id) is introduced, where S represents the user's secret information and Data id is the identifier of the data to be stored in the cloud. It generates different secret keys for different data, as symmetric cryptography is used for data encryption and decryption in the proposed cloud data storage models.

Secure data backup model for storing private and sensitive data in the cloud, utilizing attribute-based encryption.

Process begins with the data owner logging into the cloud infrastructure, sending a backup request to the cloud storage service (CSS), and receiving a data identifier in response.

Data is encrypted using a dynamically generated key and then uploaded to the private database, ensuring that each piece of data has a unique key and on retrieval, data is decrypted using the same key and its integrity is verified.

Secure data sharing one to one is for user to share data with other users in different group. Data sharing one to one process:

User shares data via CSS:

1. Request: Owner ID, storage parameter 2 (for sharing), and user ID.

2. Response: CSS sends DataID and receiver's public key.

3. Upload: Owner generates secret key, hashes/signs data, encrypts key with receiver's public key, and sends to CSS.

4. Download: Receiver decrypts secret key, then data. Checks signature, compares hashes for integrity using owner's public key.

This ensures secure one-to-one data sharing, with the owner authorizing specific users to access their data in the cloud.

Secure data sharing one to many sharing is extended to one to many, group ID is used, secret function using Sg used for shared access and group secret key is used by group for access .

Secured User's Authentication and Private Data Storage- Access Scheme in Cloud Computing Using Elliptic Curve Cryptography

(PRADEEP)

Elliptic Curve Encryption/Decryption:

Encryption Algorithm:

1. User A (Sender):

 - Chooses a positive integer 'k' randomly.

 - Has a private key 'd_A' and generates the public key '$P\_A = d\_A \times G$' (where 'G' is the base point on the elliptic curve).

2. Encryption Process:

- Message: 'P_m' (plaintext message)
- Public Key of B: 'P_B = d_B × G' (where 'd_B' is the private key of B)
- Encryption of Message:
- Calculate 'C_m = {kG, P_m + k × P_B}'
- Send 'C_m' as the encrypted message to User B.

Decryption Algorithm:

1. User B (Receiver):

- Has a private key 'd_B' and the public key 'P_B = d_B × G'.

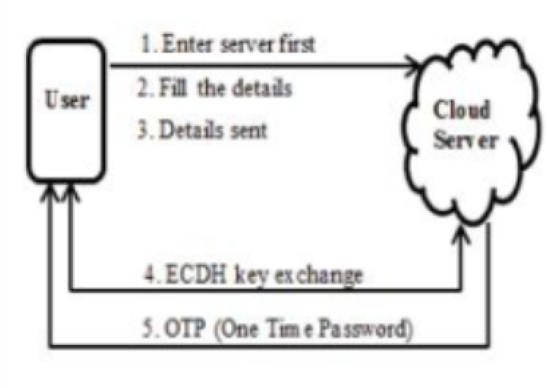
2. Decryption Process:

- Received Cipher Text: 'C_m = {kG, P_m + k × P_B}'
- Decryption Steps:
- Calculate 'P_m = P_m + k × P_B - d_B × (kG)'
- Resulting 'P_m' is the decrypted original message.

Proposed Scheme

1. Connection Establishment: The initial connection between user and cloud server is established with the help of HTTPS protocol, before the creation of account in the system.

2. Account Creation:



3. User ID Generation:

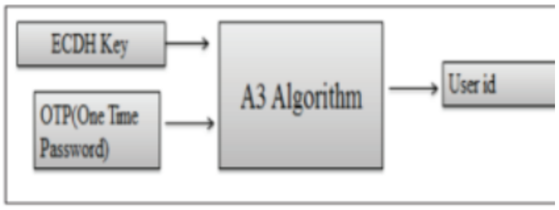


Fig.5. Input parameters for User id generation

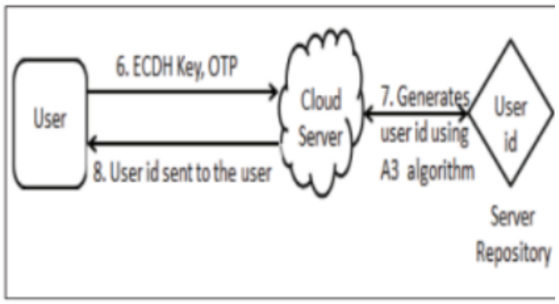

User
6. ECDH Key, OTP
8. User id sent to the user
Cloud Server
7. Generates user id using A3 algorithm
User id
Server Repository

Fig.6 User id generation

4. Authentication:

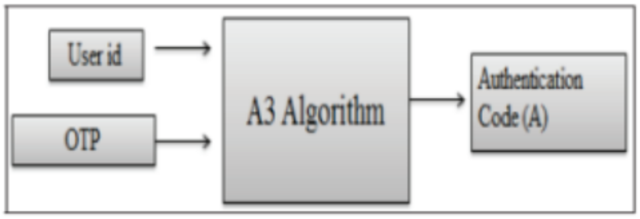


Fig.7. Authentication code generation at user end

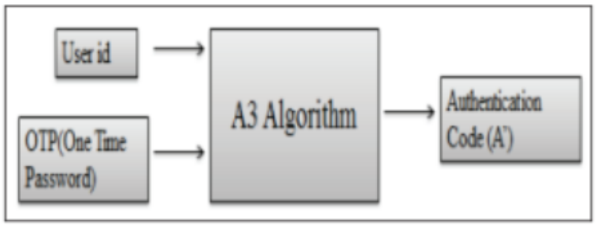


Fig.8. Authentication code generation at user end

5. Encryption and Decryption: User first encrypts the data at the client side with the help of symmetric key algorithm and encrypts the key using ECC encryption algorithm and then uploads (encrypted data and encrypted key) it to the cloud. When the user needs the data, it must firstly decrypt the key using ECC decryption algorithm to download the data and then decrypts the data with his symmetric key.

A Survey of Key Bootstrapping Protocols Based on Public Key Cryptography in the Internet of Things

(PRADEEP)

CLASSIFICATION APPROACHES OF KEY BOOTSTRAPPING PROTOCOLS:

1. Key Delivery Method:

Key Transport Mechanisms: Involve securely transferring a secret value from one party to another, often using out-of-band communication or pre-deployment. Includes one-pass, two-pass, and server-assisted approaches. Key Agreement Mechanisms: Derive a shared secret from contributions of multiple parties, resisting eavesdropping. Utilizes algorithms like Diffie-Hellman and variants, ensuring higher security against attacks like man-in-the-middle.

2. Cryptographic Primitive:

Symmetric Key Pre-distribution Schemes: Parties share a common secret key for message encryption/decryption, ensuring implicit authentication. They rely on pre-established keys but suffer from scalability and vulnerability issues. Asymmetric Key Schemes: Utilize public key cryptography (PKC) employing public-private key pairs for encryption, digital signatures, and authentication. Offer higher security but demand more computational resources.

3. Authentication Mechanism:

• Symmetric Cryptography-based Authentication: Relies on pre-shared keys or

established credentials, often using centralized servers for authentication.

• Asymmetric Cryptography-based Authentication: Involves peer-to-peer or

managed methods. Peer-to-peer methods often utilize out-of-band channels for

dynamic authentication, while managed methods rely on centralized servers.

• Other asymmetric authentication methods include identity-based authentication,

PKI, certificate-based authentication, and cryptographically generated identifiers,

each with its own security implications.

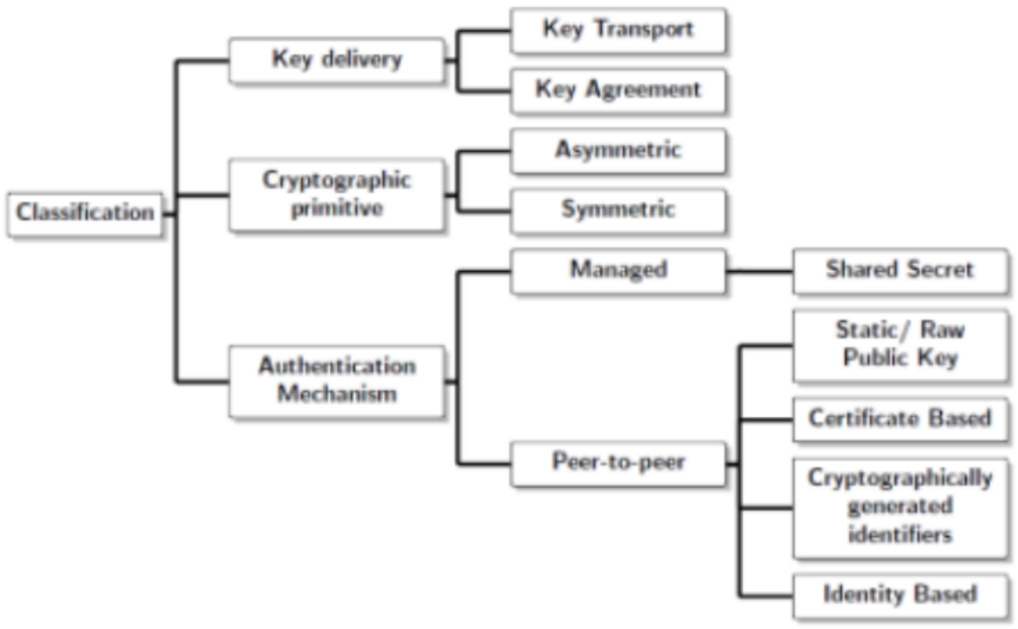


FIGURE 3. Classification approaches of key bootstrapping proposals.

ENPKESS

(ROHAN)

Reference: An efficient public key secure scheme for cloud and IoT security

Chandrasekar Thirumalai, Senthilkumar Mohan, Gautam Srivastava.

Problem Statement

The conventional RSA cryptosystem faces challenges due to increased processing time (N-bit moduli ≥2048 bits) for generating asymmetric keys.

This poses efficiency concerns of unsecure cloud computing environments and vulnerability to side-channel attacks within the Internet of Things (IoT).

Proposed Solution

The paper introduces an Efficient and Non-shareable Public Key Exponent Secure Scheme (ENPKESS) that employs a non-linear Diophantine equation, offering a threestage encryption and two-stage decryption process.

The paper applies the Knapsack method to encrypt ENPKESS keys.

Proposed Work

ESRKGS (Enhanced and Secured RSA Key Generation Scheme): ESRKGS is an RSA variant that employs four random primes, significantly increasing key size but also escalating encryption costs, with security vulnerabilities noted in certain scenarios.

Packed Knapsack: Packed Knapsack is a cryptographic approach using super-increasing integers and knapsack weights for encrypting matrix-form messages, with decryption involving the modular inverse of the knapsack weight.

Working

ENPKESS Encryption Process:

Utilizes a non-linear Diophantine equation in the key generation phase to establish an enhanced and secure RSA variant.

Implements a three-stage encryption process involving dual keys with associated polynomial equations, increasing key size for improved security.

ENPKESS Decryption Process:

Employs a two-stage decryption process with dual keys, enhancing security while minimizing memory requirements.

Integrates a packed knapsack mechanism, utilizing super-increasing integers and knapsack weights for encryption and decryption of matrix-form messages.

Security Enhancement:

Mitigates vulnerabilities associated with traditional RSA variants, addressing factors such as key size, encryption costs, and susceptibility to factorization attacks.

Achieves non-shareable public key exponent secure scheme, contributing to the robustness of the overall cryptographic system.

Packed Knapsack Encryption in ENPKESS:

The public keys α, R, N of the ENPKESS scheme are packed using the knapsack method. This involves transforming and arranging these keys based on a radix choice, resulting in a set $Ur = \{\alpha r, Rr, Nr\}$.

The maximal element $max(l)$ is determined from this set, and a set L containing the lengths of elements in Ur is constructed. These elements are then used to create an unpacked matrix P, and its reversal matrix PV in row order is generated.

To encrypt a secret message M, the packed ciphers of the matrix P are computed as $P'(n, m) = PV(n, m) * B(m, 1)$, resulting in the set $\{p'1, p'2, ..., p'n\}$. This process enhances the security of the encryption using the principles of the knapsack method.

Numerical Instance of Knapsack Encryption:

A numerical example is provided, where the input $U = \{278104404, 19, 1525114427\}$ is transformed using a radix of 9. The resulting unpacked matrix P and its reversal matrix PV are shown.

Knapsack parameters such as W, $W{-}1$, and MS are defined. The packed public key B is computed. The encryption involves obtaining the packed ciphers of P as $P'(n, m)$, which results in a set of encrypted values $\{p'1, p'2, ..., p'n\}$.

The numerical instance demonstrates the application of the knapsack encryption method in ENPKESS, ensuring a secure and efficient transformation of public keys for message encryption.