Practical Design And Implementation Of A Mobile Off-Line E-Wallet

Mbouassa Edmond Yannick [1], Hu Dong Hui [2]

[1] Department Of Computer Science, Hefei University Of Technology
Hefei , 200093 , China

[2] Department Of Computer Science, Hefei University Of Technology
Hefei , 200093 , China

Abstract

With the advancement of digital economies many mobile payment systems have seen the day, with these mobile payment systems you can make a payment to anyone who uses the same payment system as yours. But most of these systems are internet based systems. This doesn't seem to be a problem in developed countries that have permanent access to the internet. For developing countries and third world countries this is still a problem. In some areas of the world having a permanent access to internet connection is still a luxury that many cannot afford. Here we present a mobile off-line payment mobile application that can allow a user to make a payment without internet connection. However, the user will still need an internet connection to contact the bank for registration, cash withdrawal and deposit, once the cash is downloaded; users can pay or transfer cash to others without Internet connection.

Keywords: E-payment, Off-line payment, Off-line E-wallet, mobile-payment

1. Introduction

During the past twenty years, the world of science has known an exponential technological advancement, especially in computer science. Many new computer technologies have seen the day and have considerably brought changes in our lives, in the way that we communicate and treat information. We have also assisted to a huge advancement in the internet world. Internet which was something very rare not accessible for some of us, with poor content has become an essential tool for development in today's life. Nowadays many Organizations and financial institutions depend entirely on internet; Internet has also made our life much easier. Peoples communicate, reach to services and share all kind of data more easily at any time, using computers or cell phones.

During the last ten years, we have witnessed what we can call an economical revolution, with the arrival of E-commerce, the electronic money and many online payment tools. This movement has quickly gained an important place in our society and affected all forms of businesses. Countries like China have successfully demonstrated the impact of e-commerce and online payment in [1]. From shopping or selling online, to simple transactions of funds to others no matter the distance. With the huge growth of electronic commerce the world has been seriously working on secured electronic payment methods, many giants organizations like Alibaba, Apple, Tencent etc have developed mobile platforms that have completely change our way of using money, however, as efficient and popular as those platforms can be, they still inaccessible to some people in some areas of the world, especially in developing countries. This is justified by the fact that many countries in the world are still lacking an essential tool that can allow them to use the platforms. mentioned above and enjoy the benefits of electronic payment techniques: the Internet.

Statistics in [2] shows that most countries in developing countries have the lowest internet penetration rate; with only 37.3 % of penetration rate concerning countries in Africa [3].

While the rest of the world is discussing the implementation of the newly 5G technology [4], many region of the world are still struggling to implement older communication technologies, while the Internet is an easy accessible tool to people of developed countries, in other countries the internet is still a luxury that many cannot afford, even if they seem to like the idea of online payment**.** To solve this issue we came up with an idea of creating an offline electronic wallet, a platform that will allow people that can rarely access to the internet to exchange money without worrying about having internet or not.

2. Literature Review

An offline E-cash system is widely presented as a tool that offers appreciably greater security and better privacy than currently considered E-cash system with similar functionality. Most off-line E-cash systems use temper-resistant IC card which controls an E-cash issued by the card issuer. Offline E-cash system based on IC card has the threats of overspending, double spending, forgery E-cash, altering/eavesdropping transaction contents, etc. To prevent the above threats, there have been a lot of technical discussions about the

security requirements for theoretical offline E-cash protocols based on IC card [5]. David Chaum proposed the first offline e-cash system based on blind signatures in [6]. This system removes the requirement that the seller must contact the bank during every payment. Since then many electronic payment systems offering unconditional anonymity have been proposed. However, these unconditional privacy protecting systems sometimes could be misused by criminals for blackmailing or money laundering. It was proposed that anonymity revocable systems with a trusted third party involved (e.g. the government or the authorities of law enforcement), should be designed to prevent the payment systems from being misused. The concept of fair E-Cash system was put forth independently by E. Brickell [7] and M. Stadler [8] in 1995. It offers a compromise between the need of the privacy protection of users and effectively preventing the misuse by criminals. On one hand, the bank and the merchant can't obtain the identities of users by themselves. But on the other hand, under some special circumstances, with the help of the bank, the trusted third party can remove the anonymity of the transaction and recognize the user's identity.

In 1996, Y. Frankel, U. Tsiounis and M. Yung proposed a fair off-line ECash system (FOLC) [9]. G. Davida, Y. Frankel, Y. Tsiounis and M.Young improved the efficiency of FOLC system [10]. Using interactive knowledge proofs; these systems need more communication among the bank, the users and the merchants. Furthermore, both systems of J. Camenisch and G. Davida have to search in a database at the bank to be able of owner tracing. But it is not the case in our real life: The trusted third party should be responsible for revoking the anonymity rather than the bank.
In [11], which describes implementations of e-cash systems. The authors provide their own model of an e-cash scheme based on mobile trusted module architectures in which at design time, user payment tokens are composed of two modules: an untrusted but powerful execution platform (such as a smart phone) and a trusted but constrained platform (referred to as a secure element).Their scheme fulfills all common properties of e-cash while relaxing the amount of trust usually placed in the payment token. Moreover, they provide a proof-of concept implementation using commercially available platforms.
In [12] the author proposed an e-cash scheme where e-wallet carries the meaning of software and hardware system with divisible and secure e-cash used to store funds downloaded from a bank account or transfer certain amounts to other e-wallets in order to make purchases. In the general scheme, e-wallet is integrated in the entire e-money circulation system. In this system the author uses the idea to embed an observer in customer's mobile devices, through the next generation chips called physically unclonable functions (PUF)[13].

Okamoto and Ohata proposed the first ideal untraceable electronic cash [14] using the cut-and-choose methodology and introduces some basic properties, i.e., untraceability, transferability and divisibility. The cut-and-choose methodology causes low efficiency of Okamoto and Ohata's scheme. Pailles constructed a new protocol for e-cash [15] which develops the anonymity and the divisibility of the e-cash. Unfortunately, the bank has to perform a huge amount of computations. As for divisibility, Eng and Okamoto proposed a single-term divisible e-cash [16] which is not a practical divisible e-cash. The efficiency of the computation and the storage is improved in the spending protocol. Unfortunately, it does not fulfill unforgeability. In order to obtain unforgeability, Canard and Gouget proposed a divisible e-cash scheme [17].
Transferability is the other important property. Okamoto and Ohta proposed two transferable ecash systems [18, 19] which only provide weak anonymity. Later, Canard et al. proposed an anonymous transferable e-cash system [20], and analyzed the anonymity [21] in transferable e-cash.

3. Our Proposition

In this paper, we present a mobile-to-mobile off-line payment approach which can allow the user to make a payment without having to use internet connection. However, the user will still need an internet connection to contact the bank for registration, cash withdrawal and deposit, once the cash is downloaded, the user can pay or transfer cash to other users without Internet connection. This make user's to less rely on internet connection in order to make payment.

To make payment the users interact with a scan of Qr-code generated by their mobile.
To ensure the security of our framework we used elliptic curve encryption to encrypt data from end to end, we also used some techniques to avoid double spending, forgery and ensure the atomicity.
We have three entities in our model, the Bank ***B***, the Seller ***S*** and the Purchaser ***P***.
We assume that the bank is responsible for issuing the cash and responsible for securely transfer it to users; we assume that the cash issued by the bank is divisible. The seller and the purchaser have both their account in the bank.

3.1 Definitions

B :	The Bank
B_{ab} :	User's bank account balance
B_{PkA} :	Bank's private key A
B_{PkB} :	Bank's private key B
B_{PubKB} :	Bank's public key B
B_{PubKA} :	Bank's public key A

E :	Encrypted
I :	User's Information
I_n :	Randomly generated nonce for user's information encryption
M :	The user Balance inside the observer
M_n :	Randomly generated nonce for user's balance encryption
m_{sig} :	Signature of the transferring cash
m_{sn} :	Randomly generated nonce for the transferring cash encryption
N_{ref} :	Reference number
R :	The Receiver (Seller or anyone)
r :	Randomly generated number
R_{id} :	The Receiver's Identification
R_{PK} :	The Receiver's Private key
R_{PubK} :	The Receiver's Public key
S :	The Sender (Purchaser)
S_{id} :	The Sender's Identification
S_{pK} :	The Sender's Private key
S_{PubK} :	The Sender's Public key
S_{pwd} :	The Sender's password
t_1 :	Timestamp at the payment initiation time
t_2 :	Timestamp at the payment settlement time
T_m :	Transaction number (Payment)
t_{reg} :	Timestamp at the registration time
T_w :	Transfer number (Withdrawal)
t_w :	Timestamp at the withdrawal time
U_{id} :	User's Identification
U_{Pk} :	User's Private Key
U_{PubK} :	User's Public key
U_{sig} :	User's Signature
U_{sn} :	Randomly generate nonce for the user signature
W :	Cash to be withdrawn
W_n :	Randomly generate nonce for the withdrawn cash encryption
W_{sig} :	Withdrawn Cash signature
W_{sn} :	Randomly generate nonce for the withdrawn cash signature
X :	The payment message
X_n :	Randomly generate nonce for the payment message encryption

3.2 User Registration

To be able to use the platform the user must register to the bank, when registering, the bank creates an identification for the user, and also create two keys for the user, one public key and one private key. During the registration process, the user:

- enters a username which can be a pseudonym, a password and all information identifying himself, the same information on his ID (Family name, Given name, birthday, address, phone number etc…), these information are encrypted and sent to the Bank:

The bank:

- decrypts them and verify if the username and the phone number already exist in the database
- Generate an encryption pair keys for the user U_{Pk} and U_{Pubk}
- Sign the user's Id U_{id} with B_{PkA} and B_{PubkB}
- Saves the information sent by the user .
- Sends U_{sig}, U_{Pk}, U_{Pubk} to the user.

Remark: The user's information sent to the Bank's server are encrypted B_{PkB} and B_{PubkA}, present inside the Observer.

3.3 Cash Withdrawal

In this scenario the user requests the withdrawal of an amount of money to the bank; the user sends U_{id}, W, U_{sig}, U_{sn} to the Bank.

The Bank:

- Verifies that $U_{sig} = (U_{id}, t_{reg}, U_{sn})$ with B_{PkB} and B_{PubkA}
- Verifies that $W <= B_{ab}$.
- Subtract $B_{ab} - W$ and update the database
- Creates T_w and t_w
- Signs W with B_{PkA} and B_{PubkB}
- Encrypt U_{id} , W ,W_{sn}, W_{sig}, T_w , t_w with B_{PkA} and : $E(W)$
- Sends $E(W)$ and W_n to the user

The User finally:

- Decrypt $E(W)$ with U_{Pk} , B_{PubkA} and W_n
- Verifies U_{id} and W_{sig} with B_{PkB} , B_{PubkA} and W_{sn}
- Decrypts $E(M)$ with U_{Pk} , B_{PubkB} and M_n
- Add W to M $(W+ M)$
- Encrypt new M with B_{PkB} , U_{Pubk} and new M_n : $E(M)$
- Saves $E(M)$, M_n , T_w , t_w

3.4 Cash Deposit

In this scenario the user enters the amount of cash to deposit D,

- Signs D with B_{PkB} and B_{PubkA} and D_{sn}
- encrypt D, D_{sig}, D_{sn} with B_{PkB} and B_{PubkA} : $E(Q)$
- then sends U_{id}, U_{sig}, U_{sn} , $E(Q)$, Q_{sn} to the Bank.

The Bank:

- Verifies that $U_{sig} = (U_{id}, t_{reg}, U_{sn})$ with B_{PkB} and B_{PubkA}
- Decrypts $E(Q)$ with B_{PkA} , U_{Pubk} and Q_n

- Verifies that D_{sig} =(D , D_{sn}) with B_{PkA} and B_{PubkB}
- Add D to B_{ab} (D+ B_{ab})

3.5 Payment Process

The payment process consist of money exchange between two actors that both have a valid account, in this scheme we call them: Sender (the one sending money or paying money to the other user) and receiver (the one receiving or collecting money from the other user). We denote the sender with a S and the receiver with a R. This process happens sequentially in both sides. In general the transaction happens in 3 big steps:

Fig.1 payment model

Step I: on the receiver's side

The receiver initiate the payment by generating a QR code that contains the his ID R_{id}, his Public key R_{pubK}, the timestamp t_1 at the time of starting the process and the reference number for that payment N_{ref}

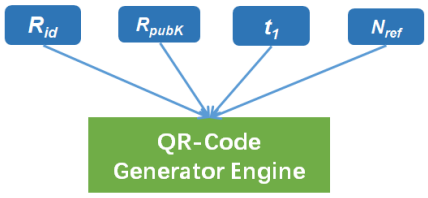

Fig. 2 Step I payment model

Step II: on the sender's side

The sender scan the QR code , then:

- Retrieves R_{id}, R_{pubK}, t_1 , and N_{ref}
- Enters the Amount of money to be transferred m and the password S_{pwd},
- The app verifies that S_{pwd} is a valid password for payment
- Decrypts $E(M)$ with S_{Pk} , B_{PubkB} and M_n : M
- Verifies that m<=M
- Subtract M - m and saves the new M
- Signs m with m_{sn} , B_{PkB} and R_{pubK} : m_{sig}
- Encrypt m with m_n , B_{PkA} and R_{Pubk} : $E(m)$
- Generate a timestamp at the moment t_2 , then create a unique transaction number T_m = (t_1 x t_2).r
  r = a random number auto generated
- The app encrypts the transaction data X , containing with X_n , S_{Pk} and R_{Pubk} : $E(X)$
  X = T_m , R_{id} , S_{id} , $E(m)$, m_n , m_{sig} , m_{sn} , t_2 , N_{ref}
- Encrypt new M with B_{PkB} , S_{Pubk} and new M_n : $E(M)$
- Saves $E(M)$, M_n , T_m , R_{id} , $E(m)$, m_n , m_{sig} , m_{sn} , t_2 , N_{ref}
- Generates a QR code readable only by receiver with $E(X)$, X_n and S_{pubK} .

Step III: on the receiver's side

The receiver scans the QR code, then :

- Decrypts $E(X)$ with R_{Pk} , S_{Pubk} and X_n : X
- Verifies the existence of the transaction N_{ref}
- Verifies that T_m doesn't alredy exist in the transaction history
  Verifies that R_{id} = U_{id}
- Decrypts $E(m)$ with R_{Pk} , B_{PubkB} and m_n : m
- Verifies that m_{sig} = $Sig(m$, $m_{sn})$ with R_{Pk} and B_{PubkB}
- Decrypts $E(M)$ with R_{Pk} , B_{PubkB} and M_n : M
- Adds m to M
- Encrypt new M with B_{PkB} , R_{Pubk} and new M_n : $E(M)$
- Saves $E(M)$, M_n , T_m , S_{id} , t_2

4. Implementation

The implementation of our offline system is subdivided in two part :

Part 1: We designed a program to use as the bank's system to simulate the interactions between the bank and the application, this program following features:

- Real-time connection with the database server
- Enrollment of new users
- Treatment of withdrawal and deposit operation
- Keys generations function
- Signature generation
- Signature verification
- Encryption function
- Decryption function.

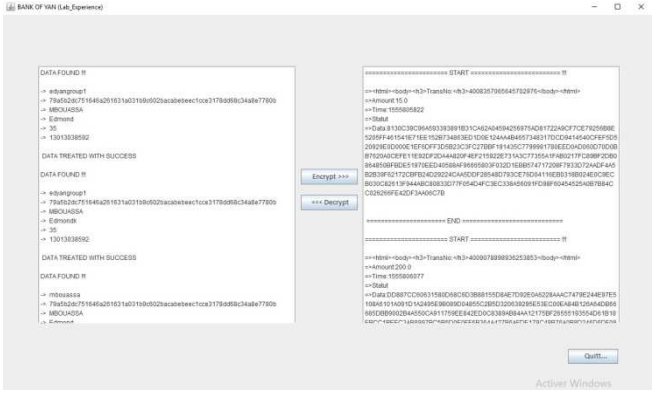

Fig. 3 bank backend service program

The program has three thread running at the same time, one thread to handle enrollment of new, one to handle the cash withdrawal and one the handle the deposit withdrawal, each of these thread has a permanent connection to the data base server, it's always checking for incoming data, if there a new data, the program retrieves the new data and start a new thread where it treats it and updates the database
This algorithm can simply be understood as follow:

START
Connect to database
Start Registration Thread
Start Withdrawal Thread
Start Deposit Thread

Registration Thread {
request new data
if (data found) {
Start registration handler Thread (*Fig. 2*)
}
}
Withdrawal Thread {
request new data
if (data found) {
Start withdrawal handler Thread (*Fig. 3*)
}
}
Deposit Thread {
request new data
if(data found) {
Start deposit handler Thread (*Fig. 6*)
}
}

Part 2: We designed the application and implemented every aspect of our payment scheme. Here is the result of our interface designs

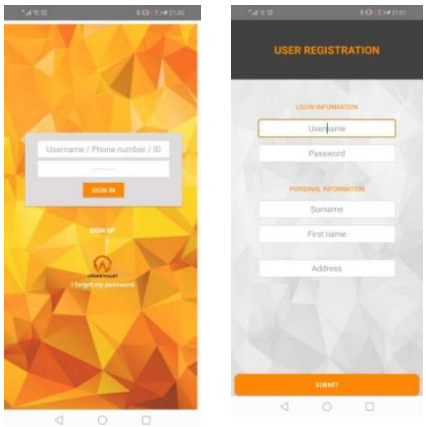

Fig. 4 Login and registration interface

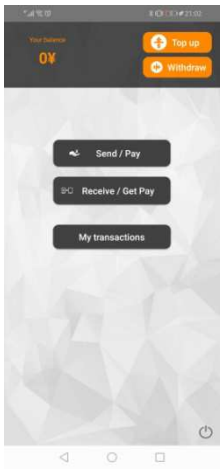

Fig. 5 Home interface

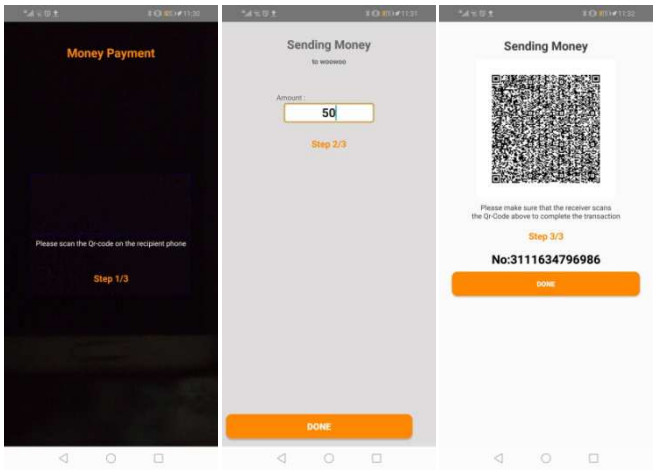

Fig. 6 Payment interface Sender's side

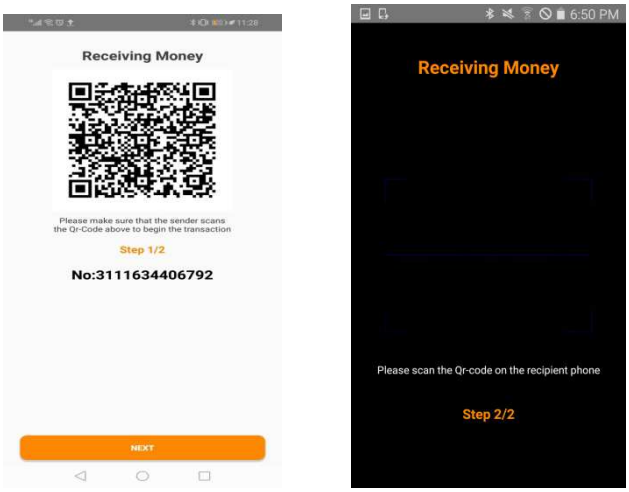

Fig. 7 Cash collection interface Receiver's side

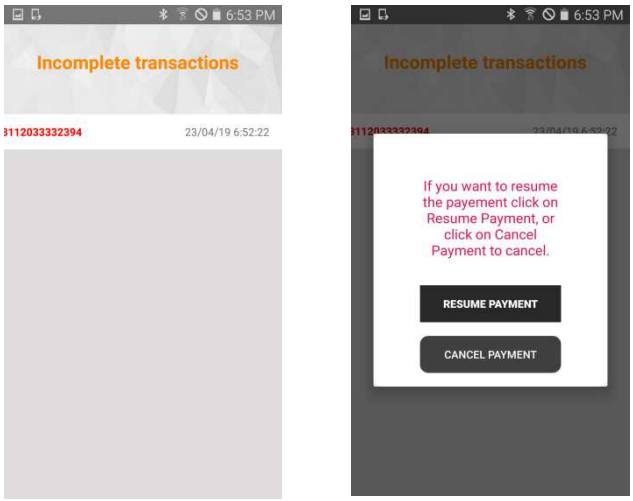

Fig. 8 Uncompleted transaction interface Receiver's side

4.1 Discussion

Case 1: Let's imagine that After scanning the Qr-code and received the cash Bob want to cheat the system by scanning again one more time, Bob gets an error message telling him that he already used the the Qr code.
How does the system handle the double spending in this case:
When bob generate the Qr-code to initiate the payment, the system generates an unique random number ***Nref*** , when Bob tries to scan Alice's payment code, the system checks the transaction number T in the database, if it doesn't exist then it checks that the reference number is the corresponding one, then he can receive the money, if it does exist then or Nref isn't the corresponding number then the payment is rejected.

Case 2 : Let Peter be a third user, trying to scan a Qr-code that wasn't meant for him. The system will directly reject the payment since Peter doesn't have the private key to decrypt the information inside the Qr-code.

Case 3: Let's imagine that Bob's phone ran out of energy before he has received the cash, the system has saved Bob's transaction as a uncompleted transaction, once Bob has restarted the phone, he can resume the payment by selecting the exact transaction on the uncomplete transaction list,. Alice on the other will select the same transaction and regenerate the Qr-code for Bob. The transaction expires after sometimes so Bob and Alice has to be quick.

5. Security And Performance

5.1 Security

In the implementation of our scheme we used an android API called Bouncy Castle, Bouncy Castle is a collection of APIs used in cryptography. It includes APIs for both the Java and the C# programming languages. The Bouncy Castle architecture consists of two main components that support the base cryptographic capabilities. These are known as the 'light-weight' API, and the Java Cryptography Extension (JCE) provider. Further components built upon the JCE provider support additional functionality, such as PGP support, S/MIME, etc.
The low-level, or 'light-weight', API is a set of APIs that implement all the underlying cryptographic algorithms. The APIs were designed to be simple enough to use if needed, but provided the basic building blocks for the JCE provider. The intent is to use the low-level API in memory constrained devices (JavaME) or when easy access to the JCE libraries is not possible (such as distribution in an applet). As the light-weight API is just Java code, the Java virtual machine (JVM) does not impose any restrictions on the operation of the code, and at early times of the Bouncy Castle history it was the only way to develop strong cryptography that was not crippled by the Jurisdiction Policy files that prevented JCE providers from performing "strong" encryption.
The JCE-compatible provider is built upon the low-level APIs. As such, the source code for the JCE provider is an example of how to implement many of the "common" crypto problems using the low-level API. Many projects have been built using the JCE provider, including an Open Source Certificate Authority EJBCA. The Bouncy Castle provider has provided support for elliptic curve since release 1.04. This was done primarily to provide support for X9.62 ECDSA. The Bouncy Castle API usage and specifications are describes in [22] and [23]

Key Generation

The interfaces representing elliptic curve keys are in the package *org.bouncycastle.jce.interfaces.*
The key pair generator function takes in two parameters: the curve name and a secure number. For the sake of experiment we used the *prime192v1* curve In our implementation. The *KeyPair* interface has a two methods, *getPublic()* and *getPrivate(),* which returns the *ECPoint* representing the public point and the private point for the elliptic curve public key and private key;

Let's ***P*** be the curve, ***r*** the secure random number, ***K*** the Key pair ***Q*** the private key and ***D*** the public key.

$$K = Gen(P, r)$$

q = K.getPrivate()
d = K.getPublic()

Encryption / Decryption

NIST has defined five modes of operation for AES and other FIPS approved block ciphers MODES [24, 25]. Each of these modes have different characteristics. We chose the AES-CTR mode for our encryption implementation. AES-CTR has many properties that make it an attractive encryption algorithm for in high-speed networking, it uses the AES block cipher to create a stream cipher. Data is encrypted and decrypted by XORing with the key stream produced by AES encrypting sequential counter block values. CTR requires an initialization vector (IV). In a block cipher, the result of encrypting a block is the input for encrypting the next block. Since the first block has no other block before it, a randomized data gets used as input. This randomized data is called the initialization vector. Using an initialization vector make the patterns in a cipher text unique such that an attacker is not able to detect patterns which may help to decrypt all or part of the original text. This specification calls for the use of a nonce for additional protection against pre-computation attacks. The nonce value need not be secret. However, the nonce must be unpredictable prior to the establishment of the IPsec security association that is making use of AES-CTR.

Let's ***M*** be the message to be encrypted, ***IV*** be the initialization vector for the encryption, ***n*** the nonce value, ***q*** the private key , ***d*** the public key, ***E*** the encrypted message and ***D*** the decrypted message.

- When encrypting we generate a secret shared key ***SK,*** then we produce the initialization vector ***IV*** and finally we encrypt the message with ***SK***, ***IV***

$$SK = GenSecret(q_1, d_2)$$
$$IV = IvSepec(n)$$
$$E = Cipher(M, SK, IV)$$

- When decrypting we generate a secret shared key ***SK,*** then we produce the initialization vector ***IV*** and finally we decrypt the message with ***SK***, ***IV***

$$SK = GenSecret(q_2, d_1)$$
$$IV = IvSepec(n)$$
$$D = Cipher(M, SK, IV)$$

Signature And Verification

Let ***M*** be the message that needs to be signed, ***IV*** be the initialization vector for the encryption, ***n*** the nonce value, ***q*** the private key , ***d*** the public key, ***E*** the encrypted message, ***D*** the decrypted message and ***Sig*** the signature.

- When signing we produce the hash of the message ***H***, then we generate a secret shared key ***SK,*** we produce the initialization vector ***IV*** and finally we encrypt the hash ***H*** with ***SK***, ***IV***

$$H = Hash(M)$$
$$SK = GenSecret(q_1, d_2)$$
$$IV = IvSepec(n)$$
$$Sig = Cipher(H, SK, IV)$$

- When verifying the signature we produce the hash of the message ***H***, we generate a secret shared key ***SK,*** we produce the initialization vector ***IV*** , we decrypt the signature ***Sig*** with ***SK***, ***IV*** finally we compare ***D*** with ***H***

$$H = Hash(M)$$
$$SK = GenSecret(q_2, d_1)$$
$$IV = IvSepec(n)$$
$$D = Cipher(Sig, SK, IV)$$

Signature is verified $\Leftrightarrow M=D$

* The hash function considered in this paper is SHA-256 algorithm

5.2 Performance

To determine the time spend by each function after the implementation, we inserted a timestamp in each function, at the beginning of the execution and the end of the execution. This procedure has been run 5 times with 2 phones with different resources capacities:

- Huawei 9 plus: Android Version 8.1.0, Processor Hisilicon Kirin 710F 2.2Ghz, RAM Memory 4GB **(1)**
- Samsung Galaxy note 3: Android Version 5.0, Processor Exynos 4 Quad 1.4Ghz, RAM Memory: 1GB **(2)**

The results are the following:

Table 1: test results of withdrawal process

Test	1	2
1	5,01	11,01
2	4,47	12,06
3	4,49	11,53
4	4,58	11,06
5	5,03	11,49

Table 2: Qr-code Generation at the initiation (Receiver's Side)

Test	1	2
1	1,56	5,04
2	2,00	5,08
3	1,57	4,58
4	2,01	5,01
5	2,01	5,03

Table 3: Reading of the Receiver's Qr-code (Sender's side)

Test	1	2
1	<1	<=1s
2	<1	<=1s
3	<1	<=1s
4	<1	<=1s
5	<1	<=1s

Table 4: Qr-code generation (Sender's side)

Test	1	2
1	2,02	7,12
2	2,00	6,57
3	2,07	6,58
4	2,01	7,06
5	2,02	7,08

Table 5: Qr-code reading (Receiver's side)

Test	1	2
1	<=1	3,02
2	<=1	3,09
3	<=1	2,58
4	<=1	3,03
5	<=1	3,03

6. Analysis And Usability

Time Average (TA)

Table 6: Qr-code reading (Receiver's side)

Process	1	2	TA
Withdrawal process	4,71	11,43	8,07
Payment: Qr-code Generation (Receiver's Side)	1,83	4,94	3,38
Payment: Receiver's Qr-code Reading (Sender's side)	<=1	<=1	<=1
Payment: Qr-code generation (Sender's side)	2,02	6,88	4,45
Payment: Sender's Qr-code reading (Receiver's side)	<=1	2,93	<=3,93
Payment process in total	<=5,85	<=7,87	12,76

The test result shows that, the mobile cell phone with a high configuration setting process treatment of data is more quick than the one with a lower one

- for the withdrawal process, the processing time average on **1** is 4,71 seconds and 11,43 seconds on **2**. This means that the average of cell phones will need approximatively 8,07 to process a withdrawal. This result is without taking the network bandwidth into account.
- for the payment process, the final results shows that, the processing time average is lower or equal to 5,85 seconds on **1** plus and 7,87 seconds on **2**. This means that the average of cell phones will need approximatively 12,76 seconds to process a payment

7. Conclusion

Electronic payment refers to the mode of payment which does not include physical cash or checks. It includes debit card, credit card, smart card, e-wallet etc.

The growing use of Electronic payment presents many advantages for our economic development. From online shopping to funds transferring E-payment as become an essential tool for users and a hot topic among researchers, government organization and high financial organizations. Among all countries in the world, half countries in the are in need of such payment system but the lack of internet connection stability or the cost of internet bills does not allow them to do so. In this paper we have demonstrate by developing an off-line e-wallet mobile software that it is possible to make payment without internet connection, of course this system has its own advantages and disadvantages like others, it is up to the user to choose which payment system is suitable for him or for its business.

References

[1] Qu, Lili and Chen, Yan, "The Impact of e-commerce on China's Economic Growth" (2014).WHICEB 2014 Proceedings. 101

[2] World Internet Users and 2019 Population Stats www. internetworldstats.com/stats.htm

[3] Internet Users Statistics for Africa 2019 www. internetworldstats.com/stats1.htm

[4] Akira Matsunaga, "Deployment Plans Toward 5G Implementation" May 24, 2017 3rd Global 5G Event in Tokyo, Japan

[5] Haeryong Park, Kilsoo Chun, Seungho Ahn The Security Requirement for off-line E-cash system based on IC Card, IEEE 10.1109/ICPADS.2005.279

[6] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Advances in Cryptology: Proceedings of CRYPTO '88, pages 319–327. Springer New York, 1990.

[7] E. F. Brickell, P. Gemmell, and D. Kravitz. Anonymity control in E-cash systems 19 July 2005 Springer

[8] J Camenisch, M Stadler Efficient group signature schemes for large groups Annual International Cryptology Conference, 410-424

[9] Frankel, Y., Tsiounis, Y., Young, M.: Fair Off-Line e-cash Made Easy. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 257–270. Springer, Heidelberg (1998)

[10] Davida, G., Frankel, Y., Tsiounis, Y., Yung, M.: Anonymity Control in E-Cash Systems. In: Luby, M., Rolim, J.D.P., Serna, M. (eds.) FC 1997. LNCS, vol. 1318, pp. 1–16. Springer, Heidelberg (1997)

[11] Scheir, M., Balasch, J., Rial, A., Preneel, B., & Verbauwhede, I. (2015). Anonymous split e-cash—Toward mobile anonymous payments. ACM Transactions on Embedded Computing Systems (TECS),14(4), 85.

[12] Eligijus S., Jonas M., Inga T (2017) Computational ressources for mobile E-Wallet System with Observers

[13] SRAM PUF, The Secure Silicon Fingerprint, White paper. Intrinsic ID, 2016. Online. [Available]: https://www.intrinsic-id.com/physical-unclonable-functions/free-white-paper-sram-puf-secure-silicon

[14] Okamoto, T. and Ohta, K. Universal electronic cash. CRYPTO 1991, LNCS 576, California, USA, 11- 15 August, pp. 324-337.

[15] Okamoto, T. and Ohta, K. Universal electronic cash. CRYPTO 1991, LNCS 576, California, USA, 11- 15 August, pp. 324-337.

[16] Chan, A. H., Frankel, Y. and Tsiounis, Y. Easy come-easy go divisible cash. EUROCRYPT 1998, LNCS 1403, Espoo, Finland, 31 May-4 June, pp. 561- 575.

[17] Canard, S., and Gouget A. Multiple denominations in E-cash with compact transaction data. FC 2010 LNCS 6052, Tenerife, Canary Islands, 25-28 January, pp. 82-97.

[18] Okamoto, T. and Ohta, K. Diposable zero-knowledge authentications and their applications to untraceable electronic cash. CRYPTO 1989, LNCS 435, California, USA, 20-24 August, pp. 481-496.

[19] Okamoto, T. and Ohta, K. Universal electronic cash. CRYPTO 1991, LNCS 576, California, USA, 11- 15 August, pp. 324-337.

[20] Canard, S., Gouget, A. and traore, J. Improvement of efficiency in (unconditional) anonymous transferable ecash. FC 2008, LNCS 5143, Cozumel, Mexico, 28-31 January, pp. 202-214.

[21] Canard, S. and Gouget A. Anonymity in transferable e-cash. ACNS 2008, LNCS 5037, NY, USA, 3-6 June, pp. 207-223.

[22] “BC-FJA (Bouncy Castle FIPS Java API) Non-Proprietary FIPS 140 -2 Cryptographic Module Security Policy” Legion of the Bouncy Castle Inc

[23] BC-FJA (Bouncy Castle FIPS Java API) User Guide Legion of the Bouncy Castle Inc.

[24] NIST, FIPS PUB 197, "Advanced Encryption Standard (AES)", November 2001.

[25] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST Special Publication 800-38A, December 2001.