Cryptography and Digital Transformation

Kazue Sako

Abstract Cryptography is implemented using discrete mathematics with security defined in complexity theory. In this article, we review some cryptographic primitives for encryption, signing messages and interactive proofs. By combining cryptographic primitives, we can design and digitally implement various services with desired features in security, privacy and fairness. We will discuss some examples such as electronic voting and cryptocurrencies.

1 Digital Transformation

Research in mathematics and cryptography play a big role in shaping our digitalized society much better in coming years. There is an immense expectation that technology on Information and Communications, known as ICT, would transform our life to be more efficient, more productive and more functional. However, these are bright side of digital transformation. We also need to take care to transform 'correctly' so that we do not suffer from unexpected consequences.

One evident characteristic of ICT is that it makes us free from physical constraints. Digital data have little weight and thus we can make thousand copies and travel thousand miles at once. While this characteristic brings benefit, it also brings threats to our life. We need alternative ways to create 'constraints' to those who is willing to harm us, and one promising approach to creating such constraints is use of cryptography.

Cryptography started as a way to conceal information. We were able to design cryptographic algorithm that is computationally infeasible to recover the message without knowledge of a decryption key. There are rigorous mathematical proofs that guarantee that indeed this characteristic holds based on some hard problems, like NP problems or factorization. So this computational difficulty would serve as an alternative constraints in a digital world.

K. Sako (✉)
Waseda University, Tokyo, Japan
e-mail: kazuesako@aoni.waseda.jp

T. Chacón Rebollo et al. (eds.), *Recent Advances in Industrial and Applied Mathematics*, ICIAM 2019 SEMA SIMAI Springer Series 1,
https://doi.org/10.1007/978-3-030-86236-7_9

In this article, we provide two examples of use cryptography to implement secure digital systems. One is digitalization of voting system, and the other is digitalization of payment system called Bitcoin. Prior to these two examples we oversee some cryptographic primitives such as encryption schemes, digital signature schemes and interactive proofs.

2 Cryptographic Foundations

In this section, we will introduce three fundamental notions in cryptography. They are Encryption Schemes, Digital Signature Schemes and Interactive Proofs.

2.1 Encryption Schemes

First, we begin by introducing two types of encryption schemes, depending on how we use keys. The first type, which is called Symmetric-key encryption schemes, uses the same key for both encryption and decryption. This type of encryption schemes existed since the age of Gaius Julius Caesar. The new type of encryption is called Publickey encryption schemes or Asymmetric-key encryption schemes, where we use different keys for encryption and decryption. Moreover, the key to encrypt data can be made public (Fig. 1).

Let us briefly discuss some mathematical model to define encryption schemes and its security. Encryption schemes, either symmetric or asymmetric, can be mod-

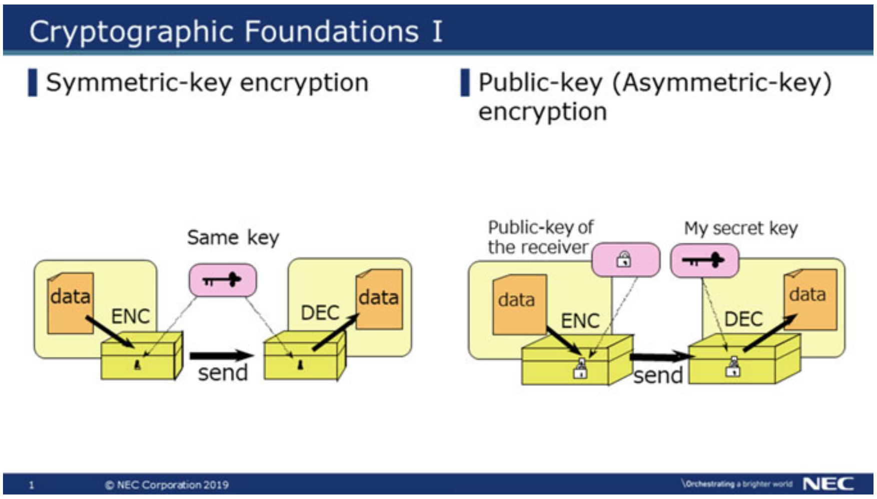

Cryptographic Foundations I
Symmetric-key encryption
Public-key (Asymmetric-key) encryption
Same key
data
ENC
DEC
data
send
Public-key of the receiver
My secret key
data
ENC
DEC
data
send
1
© NEC Corporation 2019
\Orchestrating a brighter world
NEC

Fig. 1 Two types of encryption schemes

eled in three non-deterministic functions, namely KeyGeneration, Encryption and Decryption, with a security parameter k. KeyGeneration, on input k, outputs a key pair EncKey and DecKey. (In case of Symmetric Key encryption schemes, EncKey = DecKey holds.) Encryption Function, given a message m from its domain and EncKey, outputs a ciphertext c.

$$c = \text{Encryption}(k, m, \text{EncKey})$$

Similarly, Decryption Function, given a ciphertext c from its domain and DecKey, outputs a message m'.

$$m' = \text{Decryption}(k, c, \text{DecKey})$$

A triplet of nondeterministic functions (KeyGeneration, Encryption, Decryption) is called Encryption scheme if and only if: For any k, for any output (EncKey, DecKey) of KeyGeneration on input k, and for any message in m,

$$m = \text{Decryption}(k, \text{Encryption}(k, m, \text{EncKey}), \text{DecKey})$$

holds.

As seen in the definition, even an Encryption function that returns m as c is an Encryption Scheme. So we need to define what property we need to call an Encryption Scheme secure. Cryptographers had studied various ways to do this. A fundamental observation is: given any two messages m_1 and m_2, and given any ciphertext c_i of either m_1 or m_2, the encryption scheme is secure if no one can guess to which message a ciphertext c decrypts to with probability more than half. To be more rigorous, we need to define this in an asymptotic manner. That is, if we chose large enough k, the probability of guessing can be made larger than $1/2 + \epsilon$. We note that in Asymmetric Encryption Schemes, guessing is hard even if they know EncKey that was used to create c. There are various other security definitions for Encryption Schemes, be it strong or weak [1].

To prove security of some concrete Encryption Schemes, we assume existence of some one-way functions or some difficult problems like factorization.

2.2 *Digital Signature Schemes*

Another exciting tools related to Public Key Encryption Schemes are Digital Signature Schemes. If we can have two related keys PubKey and PrivKey, where one can publish PubKey without worrying about secrecy of PrivKey, we can construct a scheme that serves as Digital Signatures. A person would sign a message with PrivKey and outputs a signature sig. Anyone can verify whether or not the signature was generated using a key corresponding to PubKey, by performing Verification (Fig. 2).

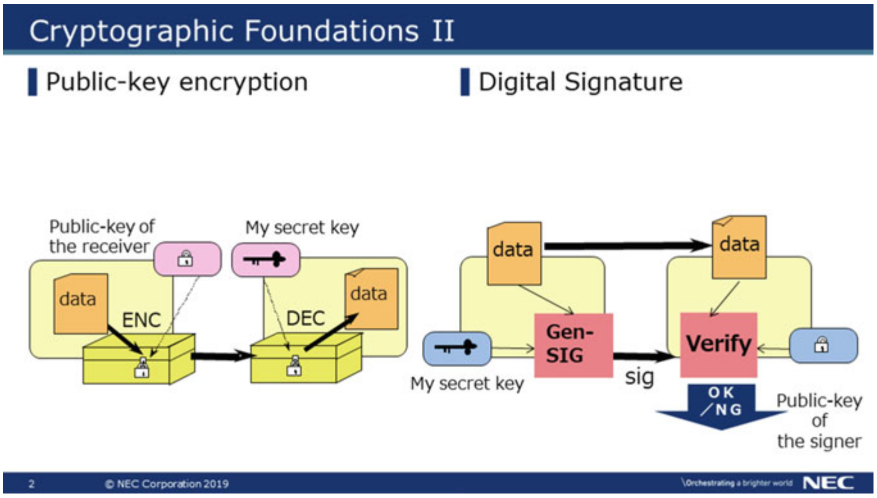


Fig. 2 Digital signature schemes

Similarly, Digital Signature Scheme is modeled by three nondeterministic functions (KeyGen, Gen-SIG, Verify). KeyGen, on input security parameter k, outputs a key pair PrivKey and PubKey. Gen-SIG Function, given a message m from its domain and PrivKey, outputs a signature sig.

$$\text{sig} = \text{Gen-SIG}(k, m, \text{PrivKey})$$

Verify Function, given a signature sig from its domain, the message m and PubKey, outputs either OK or NG.

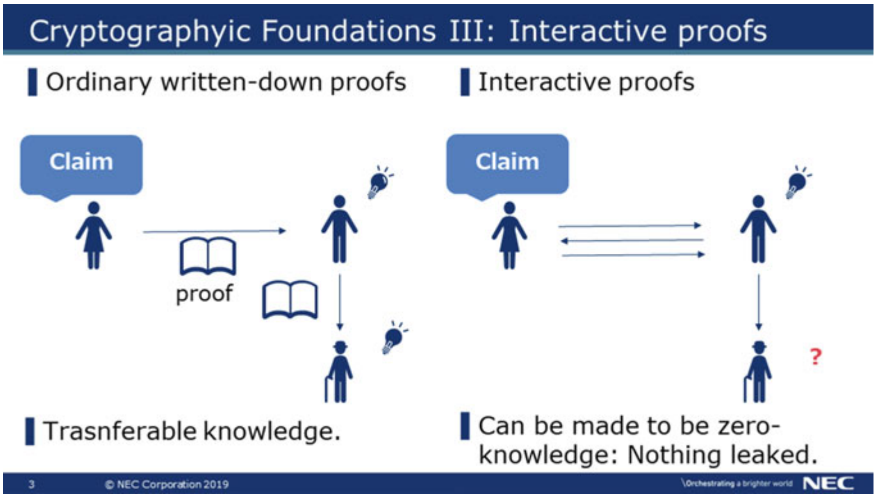
$$\text{OK/NG} = \text{Verify}(k, \text{sig}, m, \text{PubKey})$$

A triplet of nondeterministic functions (KeyGen, Gen-SIG, Verify) is called Signature scheme if and only if: For any k, for any output (PrivKey, PubKey) of KeyGeneration on input k, and for any message in m,

$$\text{OK} = \text{Verify}(k, \text{Gen-SIG}(k, m, \text{PrivKey}), m, \text{PubKey})$$

holds.

For security of signature schemes, we want to claim that it is only a person who knows PrivKey can generate sig corresponding to m that the Verify Function outputs OK. For this purpose, we claim a Signature Scheme is secure if there is an algorithm that can generate signatures that Verify outputs OK, then we can use the algorithm to 'extract' PrivKey. For sake of space, please refer to reference [1] for more mathematical definition for security of digital signature schemes.



Cryptographyic Foundations III: Interactive proofs

- Ordinary written-down proofs
- Interactive proofs

- Trasnferable knowledge.
- Can be made to be zero-knowledge: Nothing leaked.

Fig. 3 Interactive proofs

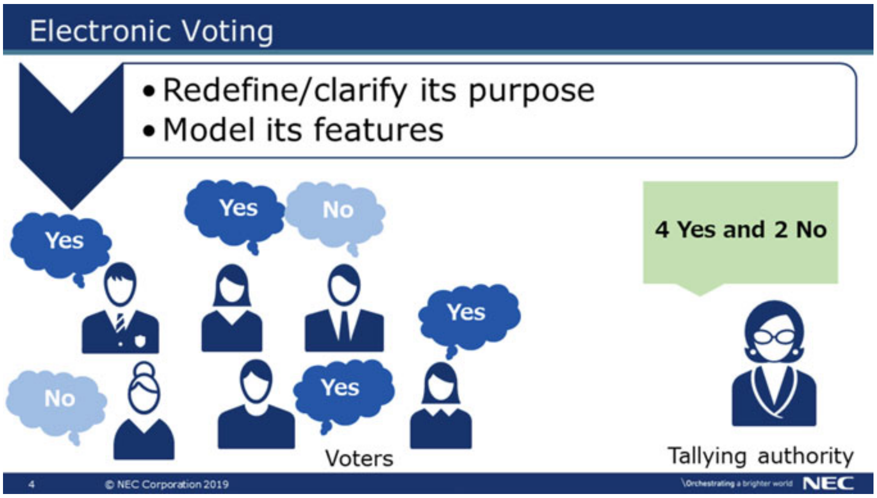
2.3 Interactive Proofs

The last primitive we will discuss in the article is Interactive Proofs. In Mathematics, when we say Proof, it is usually something that can be written down in the paper and those who have seen the Proof can verify the correctness of its claim. So the script of Proof is non-interactive. The Prover alone would generate the script of Proof by himself. Also the script of Proof is transferable, that any party who have seen the Proof can verify that the claim is correct.

Instead, there are protocols where Prover and Verifier talks interactively and at the end Verifier is persuaded that the Claim is correct. This is called Interactive Proofs (Fig. 3). This type of interactive proofs can provide further characteristic that the Verifier learn nothing from the interaction except that the Claim is correct. That is, Verifier learned no knowledge or zero knowledge in engaging the proof protocol. These types of protocols are called Zero Knowledge Interactive Proofs, which are frequently used in cryptographic protocols. Because the Verifier learned no new knowledge, he cannot prove to a third party that the Claim Prover proved is correct.

3 Digitalizing Voting

In this section we discuss how voting procedure can be securely digitalized using cryptography. Typically the process of designing cryptographic protocols consists of clarifying the purpose and modeling its feature, then design the protocol, and verify the designed protocol meets the previously set goal.



Electronic Voting

- Redefine/clarify its purpose
- Model its features

Fig. 4 Model of electronic voting

3.1 Requirements for Voting

So let us clarify the purpose of the voting and its desired property. Here, we assume there is a list of legitimate voters with their respective public keys and a Tallying authority. Each legitimate voter cast either yes or no vote and the Tally authority wants to have a correct counting of the votes (Fig. 4). The three main requirements we need to meet are the following:

1. Only legitimate voters vote, and one vote per voter.
2. Tallying authority cannot announce faulty results.
3. No one can learn how each voter voted.

3.2 Designing Voting Protocol

It seems these three requirements are hard to achieve simultaneously. If we let all legitimate voters sign their vote, then the first requirement can be met. However, if the votes are signed with the voter's key, it means the votes are not anonymous thus conflicts the third requirement. If we make all votes anonymous, then we cannot verify if the votes are from legitimate voters or even if they are, they could have voted more than once. Moreover, we cannot verify if the Tallying Authority just neglected some of the anonymous votes cast in counting the tally.

There are several ideas to meet all three requirements that seems conflicting. In this subsection, we will discuss one of such ideas using shuffling [2].

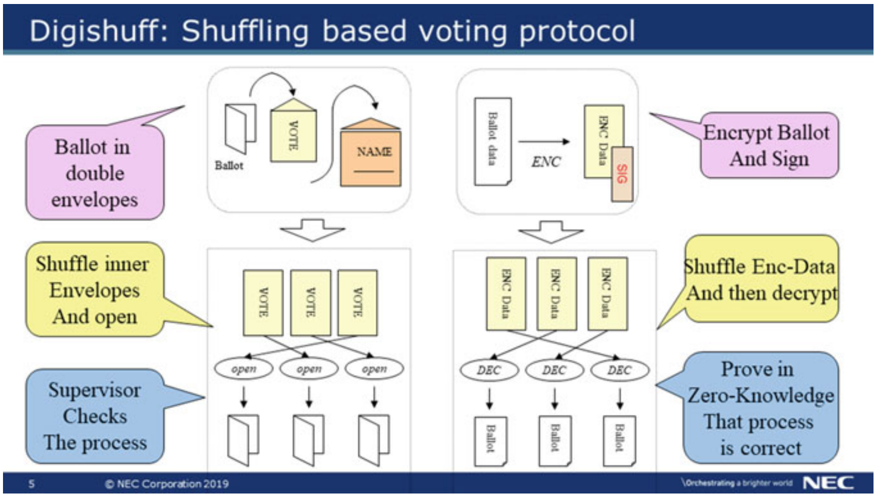
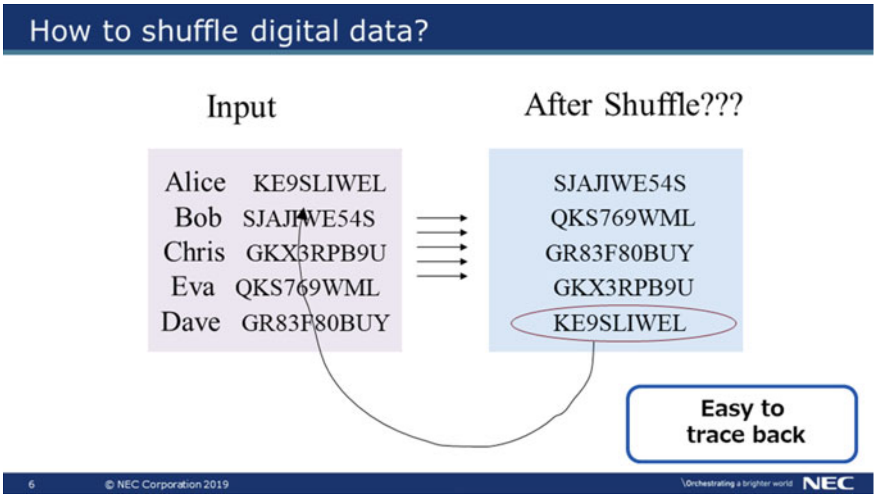

Digishuff: Shuffling based voting protocol
Ballot in double envelopes
Ballot
VOTE
NAME
Ballot data
ENC
ENC Data
SIG
Encrypt Ballot And Sign
Shuffle inner Envelopes And open
VOTE
open
Shuffle Enc-Data And then decrypt
DEC
Ballot
Supervisor Checks The process
Prove in Zero-Knowledge That process is correct
© NEC Corporation 2019
\Orchestrating a brighter world
NEC

Fig. 5 Overview of voting protocol using shuffling

The underlying idea came from how we meet those requirements using paper ballots in voting. In one providence, a voter fills in his paper ballot and put in a blank envelope. Then the voter puts this bank envelope in a larger envelope and signs with the voter's name. The voter hands this envelope to the Tallying Authority. The Tallying Authority can verify that the voter is a legitimate voter and has hand in one envelope, but because they are in an envelope the Authority cannot learn the vote. How about counting? On the day of counting the votes, all the outer envelopes are removed, but still in a blank inner envelope. All blank envelopes are thrown on the table and the envelopes will be shuffled manually so that no one learns which inner envelope came from which outer envelope. After adequate shuffling are performed, inner envelopes will be opened and count the ballots within. All the procedure will be supervised by an observer so that Tallying Authority cannot cheat while shuffling or opening the envelopes. So this trick may be able to use in digitalization (Fig. 5).

So we will encrypt the ballot using a public key of the system to mimic the blank inner envelope. As an outer envelope, the voters would sign on the encrypted ballot, and cast to the Tallying Authority. The Authority learns from the signature on the encrypted ballot that the ballot is from a legitimate voter and the same voter had not voted more than once, but the ballot itself cannot be seen as it is encrypted. Then the Authority removes the digital signature part and 'shuffles' the encrypted ballots. After the encrypted ballots has been well mixed, that is, it has been made difficult to match who submitted the encrypted ballot, the ballots will be decrypted to enable tallying. This way, we can ensure that we have only counted legitimate voter's vote once, and authority would not learn the vote of each voter as long as decrypting keys are kept safe. To ensure that the Authority performed correct Tallying, the Authority would provide Zero Knowledge Interactive Proofs to prove that it has


How to shuffle digital data?

Input

Alice	KE9SLIWEL
Bob	SJAJIWE54S
Chris	GKX3RPB9U
Eva	QKS769WML
Dave	GR83F80BUY

After Shuffle???

SJAJIWE54S
QKS769WML
GR83F80BUY
GKX3RPB9U
KE9SLIWEL

Easy to trace back

6 © NEC Corporation 2019 \Orchestrating a brighter world NEC

Fig. 6 Permutation is not shuffling

followed the procedure correctly and that the result of the tally is trustworthy. In the next subsection, we discuss in more detail how we 'shuffle' digital data.

3.3 *Shuffling Encrypted Data Using Probabilistic Encryption*

If 'shuffling digital data' was simply changing the location of some digital data, then even after shuffling it is easy to spot which digital data came from whom, by matching the bit patterns (Fig. 6).

So in digital shuffling, we need to change a look of digital data. For this purpose, we are going to use a public key encryption scheme that is probabilistic [3]. That is, the encryption function is non-deterministic, therefore there are many ciphertexts that decrypt to a same message. So changing 'the look' of encrypted digital data is to replace the encrypted data with another encrypted data that decrypts to the same message. Figure 7 illustrates such shuffling procedure. First a list of encrypted ballots are permutated. Then each encrypted ballot is replaced with another encrypted data without changing the content of the ballot. Looking at the input list and the output list, it is difficult to trace which ballot was shuffled to which position.

An example of a probabilistic encryption scheme that offer this characteristic is called ElGamal Encryption [4]. Here we provide an overview of the scheme. ElGamal Encryption is based on the assumption that given a prime p, an generator g of Zp and $y = g^a \bmod p$, it is difficult to compute a from (p, g, y) for randomly chosen y in Zp. This is called Discrete Logarithm Problem. So KeyGeneration function for ElGamal Encryption is generating p of length k (security parameter) g, and y for

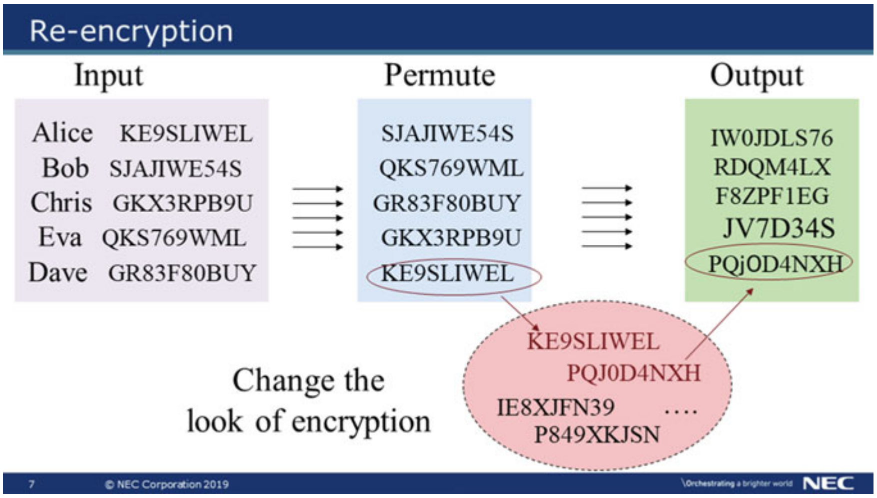


Fig. 7 Shuffling procedure

randomly chosen a. Public Key will be (p, g, y) and the exponent a will serve as secret key. Encryption function, on input message m in Zp and Public Key (p, g, y), generates a random number r, and outputs

$$(c_1, c_2) = (g^r \bmod p, m * y^r \bmod p)$$

as a ciphertext of m. On input $(c1, c2)$ and secret key a, Decryption function performs $c2/(c1)^a \bmod p$ which should be equal to the message m if the ciphertext was correctly conveyed. In order to change the look of $(c1, c2)$,

$$(d_1, d_2) = (c_1 * g^s \bmod p, c_2 * y^s \bmod p)$$

for a randomly chosen s, would provide another different looking ciphertext that also decrypts to the message m. It is interesting to see that this transformation can be performed without the knowledge of the secret key.

4 Bitcoin Blockchain

Perhaps one of the most impressive digital transformation through cryptography was digitalizing 'money' called Bitcoin [5]. There are many prepaid electronic money systems today like PayPay, but it is restricted to one currency and there is an accountable organization who is operating the system. Satoshi Nakamoto designed a system where only the algorithms ensure the correctness of the money transfer and excluding the existence of a centralized authority. We provide below an overview of his design. We note some details are omitted for the sake of simplicity.

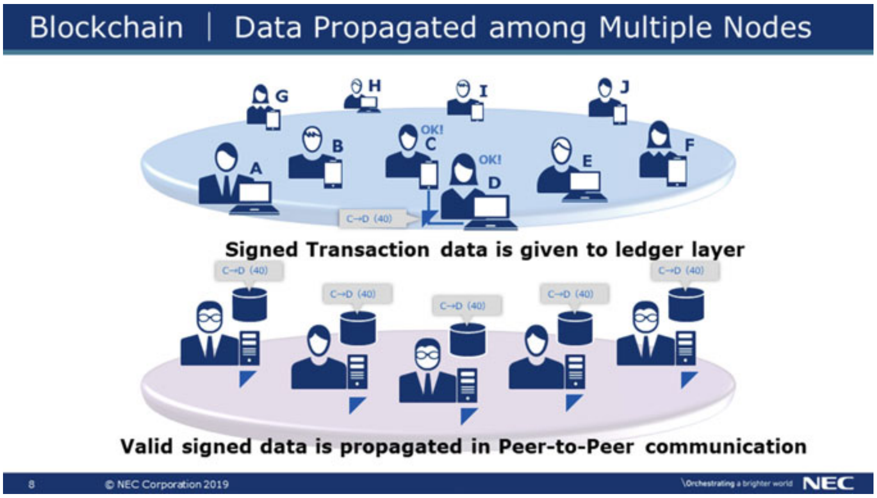
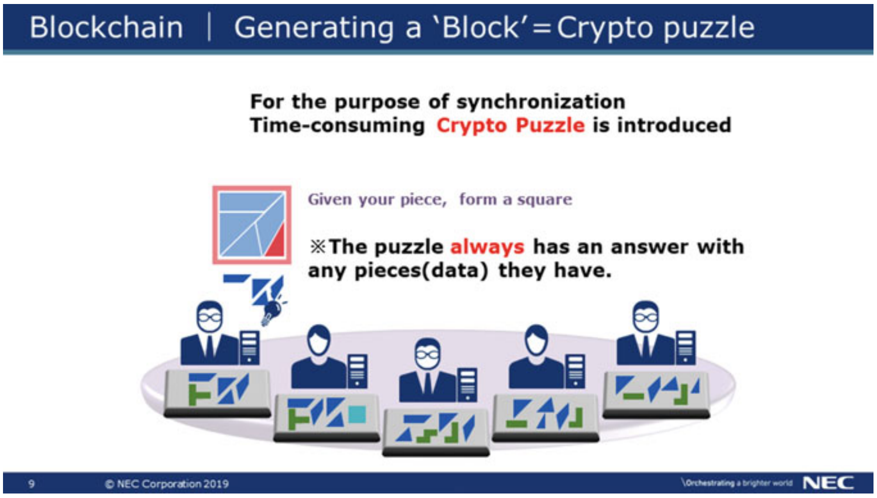

Blockchain | Data Propagated among Multiple Nodes
G
H
I
J
A
B
OK!
C
OK!
D
E
F
C→D (40)
Signed Transaction data is given to ledger layer
C→D (40)
C→D (40)
C→D (40)
C→D (40)
C→D (40)
Valid signed data is propagated in Peer-to-Peer communication
8
© NEC Corporation 2019
\Orchestrating a brighter world
NEC

Fig. 8 Data managers and transaction logs

4.1 Modeling Blockchain

Blockchain is a technology that is used to manage transaction data in Bitcoin. There are users of Bitcoin who issue transaction data, typically saying ‘sending x Bitcoin from my account yyy to the address zzz.’ The transaction is accepted if the message is indeed sent from the owner of the account yyy and indeed there are x Bitcoin left in the account. The log of transaction infers that after the transaction has been accepted, x Bitcoin should be decreased from the account yyy and added to the account zzz. Unlike previous systems where there is one organization keeping record of all the transactions, there are multiple voluntary ‘Data managers’ in Bitcoin known as Full Node, connected in Peer-to-peer fashion. When a user issued a transaction, Data managers check its correctness and propagates the transaction to other Data managers. The ideal goal is that all the Data managers keep these transaction log in a consistent way (Fig. 8). However, as transaction logs are created by various account holders internet-wide and that communication through Peer-to-peer network may not always be perfect, there is no guarantee that the list of logs are consistent among all the Data managers. So the big problem Satoshi had to solve was how to synchronize the transaction log among the Data managers while they are connected in asynchronous Peer-to-peer network.


Blockchain | Generating a 'Block' = Crypto puzzle

For the purpose of synchronization
Time-consuming Crypto Puzzle is introduced

Given your piece, form a square

※The puzzle always has an answer with any pieces(data) they have.

9 © NEC Corporation 2019 \Orchestrating a brighter world NEC

Fig. 9 Crypto puzzles for synchronization

4.2 Crypto Puzzle for Synchronization

A core idea behind synchronization is to restrict frequent distribution of transactions. If the distribution happens infrequently, for example once in every 10 min or so, that should provide enough time within Peer-to-peer network to share the same data. In order to achieve this, Bitcoin blockchain is designed so that a bulk of transaction log are bundled in a block, and the block cannot be distributed among Data Managers unless accompanied by a certain solved crypto puzzle related to the content of that block. This crypto puzzle is so designed that the puzzle for any block can be solved with high probability, but is time consuming. We note that while the puzzle is hard to solve, it is easy for other Data managers to verify that the solution is correct (Fig. 9).

In order to define crypto puzzle, we use a mathematical function called Hash Function. Hash Function deterministically maps an arbitrarily long input string to a fixed length integer of say 256 bits. The output is called a hashed value. With cryptographically secure hash function, it is computationally difficult to find two different input that maps to a same hashed value. There are known algorithms that is believed to achieve this property, such as SHA-256 [6].

Let us assume a Data Manager wants to add bulk of data $D_1, \ldots, D_n$, on top of the latest Block data Bn. The Puzzle is defined to find an string str that satisfies the following equation.

$$\mathrm{Hash}(\mathrm{Hash}(\mathrm{Bn}) \parallel D_1 \parallel \ldots \parallel D_n \parallel \ \mathrm{str}) < 2^{\mathrm{Bn}(k)}$$

where || represents concatenation of strings and $\mathrm{Bn}(k)$ is an integer defined from the previous block Bn, which is called difficulty. A typical output of Hash function is an integer of length 256, so if $\mathrm{Bn}(k)$ is about 60, one need to try many possible str to check if it meets the equation. The difficulty is so designed that this try and error process would take 10 min on average to find the desired string str.

The list of Data $D_1, \ldots, D_n$, accompanied by the correct puzzle solution str, is the propagated as a new block within Data Managers. Other Data Managers who received the block verifies the correctness of the solution. If correct, they add this block on top of the previous blocks, as the chain of data store. Then they will try to solve the next puzzle based on the new block with other transaction log that has not yet been stored in the blockchain.

4.3 Incentives for Data Managers

We conclude the overview of Bitcon Blockchain by mentioning why the Data managers spend their computational effort to solve meaningless puzzle. The Data managers are awarded by Bitcoin if they solved the puzzle and followed by the future Blocks. Their incentives for receiving the award play a central role in maintaining consistent data among Data managers, and distract them from behaving maliciously.

5 Concluding Remarks

In this article we have discussed some of the examples of securely implementing current social activities in cyber world using cryptography. We have shown some of the cryptographic primitives are defined mathematically. The procedure to design secure protocols begin with clarifying the goal and requirements and then design to meet those criteria. Although these examples show that cryptography is a promising approach, we still lack in technology to model and evaluate mathematically overall system for digital transformation. The author sincerely hope that this article would encourage the researchers in mathematics, cryptography and information technology to get together and share their strengths for the goal of making our digital society more secure and fair place.

References

1. Goldreich, O.: Foundation of Cryptography. Cambridge University Press (2009)
2. Furukawa J, Mori K, Sako K (2010) An implementation of a mix-net based network voting scheme and its use in a private organization. In: Towards Trustworthy Elections, pp. 141–154 (2010)
3. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)

4. El Gamal, Taher: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory **31**(4), 469–472 (1985)
5. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2009). https://bitcoin.org/bitcoin.pdf
6. NIST FIPS 180-4: Secure Hash Standard (SHS)