BlockPGP: A Blockchain-based Framework for PGP Key Servers

Alexander Yakubov *, Wazen M. Shbair *, Radu State *
* University of Luxembourg, SnT, 29, Avenue J.F Kennedy, L-1855 Luxembourg
Email:{alexander.yakubov, wazen.shbair, radu.state}@uni.lu

***Abstract*—Pretty Good Privacy (PGP) is one of the most prominent cryptographic standards, offering end-to-end encryption for email messages and other sensitive information. PGP allows to verify the identity of the correspondent in information exchange as well as the information integrity. It implements asymmetric encryption with certificates shared through a network of PGP key servers. Many recent breaches show that certificate infrastructure can be compromised as well as exposed to operational errors. In this paper, we propose a new PGP management framework with the key server infrastructure implemented using blockchain technology. Our framework resolves some problems of PGP key servers focusing in particular on fast propagation of certificate revocation among key servers and elimination of man-in-the-middle risk. We also provided user access right control where only the certificate holder can change information related to the certificate. We designed and developed a prototype for key server deployment on permissioned Ethereum blockchain. Permissioned blockchain should allow to control the costs of PGP key server infrastructure maintenance at the present level.**

***Index Terms*—PGP; PKI; Key Server; Blockchain**

I. Introduction

Pretty Good Privacy (PGP) is popular cryptographic protocol used for encryption of e-mail messages, text and files, directories and even disk partitions. PGP concept is based on asymmetric public key cryptography for user identification, where each user is given public and private keys. To send confidential information, the sender firstly retrieves the recipient public key, then the sensitive information (e.g. email message) is encrypted with temporary symmetric key and the sender encrypts the symmetric key with the receiver's public key and adds the encrypted symmetric key to the message. As the public key is derived from the private key, but not vice versa, only the private key holder is able to decrypt and access the message [1]. The distribution and management of PGP key pair is one of the main issues that need to be solved.

In contrast to traditional centralized hierarchical Public Key Infrastructure (PKI) used for issuing SSL/TLS certificates by Certificate Authority (CA), PGP adopts distributed trust network for certificate identity confirmation known as "Web of Trust". There is no central authority which everybody trusts, but instead, participants sign each other's keys and progressively build a web of individual public keys interconnected by links formed by this signatures.

Although PGP benefits from absence of centralized CAs as points of failure in PKI due to PGP's distributed architecture, the validity assessment of certification paths can be expensive with the growth of participant amount [2]. Moreover, recent research [3] disclosed a major breach in some mail clients using PGP, allowing to retrieve decrypted body of a protected email message. Although OpenPGP as a concept remains intact, the mail clients should be updated to remove the vulnerability. There are also potential problems with PGP key servers including Man-in-the-Middle attack. According to description of PGP key server protocol[1] key servers presently are not considered as a secure method of public key distribution.

A. Our contribution

The contribution of this paper is threefold:

1) We designed blockchain-based framework (BlockPGP) providing reliable management for OpenPGP certificates and key server infrastructure. Based on implementation of this framework PGP key server security challenges mentioned below can be solved.
2) We proposed technique of blockchain-related data incorporation into PGP certificate in line with the current OpenPGP standard. Putting blockchain-related data to PGP certificate is crucial for arrangement of user access control in a key server.
3) We developed a prototype that demonstrates the practical applicability of the proposed approach. The prototype is distributed and available on Github[2].

B. Web of Trust (WoT)

WoT is a decentralized public key environment where each participant of the ecosystem can "introduce" public keys of other participants. It substantially differs from the centralized hierarchical concept of traditional SSL certificates also known as PKI. In PKI a certificate can be introduced only by CA, a participant with a special status, while in PGP any participant can be considered as a certificate authority from PKI viewpoint. PGP users are able to designate public keys of others with different levels of trust indicating how trustworthy the signature (introduction) of the certificate holder when he signs public key certificates of other participant. In other words, how trustworthy his introduction of other participants is.

There are four levels of trust in PGP regarding the introduction of other certificates:

[1]https://tools.ietf.org/html/draft-shaw-openpgp-hkp-00
[2]https://github.com/alyakubov/blockpgp

- ***Full*** **(level=4)**: certificate holder's signature of other users' certificates is fully trusted
- ***Marginal*** **(level=3)**: certificate holder's signature can be trusted, but it is better to find other signatures with full trust to confirm the introduction of a certain certificate
- ***Untrustworthy*** **(level=2)**: certificate holder's signature should not be trusted and his signature on other users' certificates should be ignored
- ***Don't know*** **(level=1)**: there is uncertainty about trustworthiness of certificate holder's signatures of other users' certificates

Based on the multiple introductions of PGP certificates by different users it is possible to create a "chain of trust", or a path from one user to another when identity confirmation is required for sending protected information. Basically, this problem can be considered as finding paths between two vertices in a directed graph with variable vertix weights. To reach a reasonable identification trust level for a given third-party public key one should find several paths with number of edges less than five ([4], Chapter 3). Notably trust levels are stored locally in GNU OpenPGP application's data (gpg) and, to best of our knowledge, are not loaded to PGP key servers discussed below. In our approach we will not use trust level data of PGP participants.

The main advantage of PGP's WoT public key infrastructure stems from its distributed nature as it excludes any central point of failure. However, this implies difficulty for new or remote users to join the network, since some existing member of WoT typically must meet with the new user personally to have her/his identity verified and public key signed for the first time (usually it is done on PGP *Keysigning events*[3] arranged for authentication of new participants).

C. Public Key Infrastructure (PKI)

In contrast to PGP with its distributed introduction of certificates known as WoT, PKI is a hierarchical structure of certificate issuers (Certificate Authorities) authenticating identities over Internet. The PKI defines the certificate issue and management policies which for instance envisage periodic audits of CAa s as well as other security measures [5]. PKI management of public keys is based on the certificate standard X.509 defined as a data structure that binds public key values to subjects (for instance domain names). The binding is asserted by CAs digitally signing each certificate. The CA may base this assertion on profound validation of the private certificate holder's identity [6].

There are three types of X.509 certificates according to the depth of verification: the Domain Validated certificate (DV) asserts that a domain name is mapped to the correct web server (IP address) through DNS. The Organization Validated certificate (OV) also envisages additional CA-verified information, such as an organization name and a postal address. These extra validations make OV SSL Certificates more expensive compared to DV certificates. Extended Validation certificate (EV) implies the highest level of authentication, including diligent human validation of a site's identity and business registration details. Because of the extensive validation, EV is the most expensive among digital certificates [7].

[3]https://www.debian.org/events/keysigning

D. Security issues of PGP and PKI

Both PGP and PKI should provide fast update and revocation of certificates efficiently handling risks associated with lost or compromised private keys, etc. Thus most vulnerable point may be infrastructure of PKI's Certificate Authorities being central point of failure.

Recent research [3], however, disclosed a major breach in some mail clients using PGP allowing to retrieve decrypted body of a protected email message with manipulations of HTML tags built-in to the message. Although this vulnerability can be resolved by careful use of the client software with PGP support, still there are many questions regarding PGP security. For instance, there are potential problems with PGP key servers including Man-in-the-Middle attack stemming from potential breaches of PKI used in user's exchange with key servers. Details regarding PGP key servers' security problems will be discussed later.

The security of PKI systems relies on the infrastructure of CAs. To resolve CAs' risks as single centers of failure, it is possible to impose strict technical and management requirements (for instance the audits we discussed above). However, many breaches show that the security of CAs can be compromised or subjected to operational errors. As CAs are designed to be crucial for PKI's security, CA's errors or mismanagement have resulted in unauthorized certificates being issued [8]. For instance, one of well known events is the security breach of CA DigiNotar, which led to use of the company's infrastructure for the issue of hundreds of rogue digital certificates for high-profile domains, including those for Google.com and Facebook.com. These certificates were later used in a mass surveillance attack against some Internet users [9].

In another PKI breach, a Malaysian subordinate certificate authority DigiCert Sendirian Berhad (Sdn. Bhd.) mistakenly issued 22 weak SSL certificates, which could be used to impersonate websites and sign malicious software. As a result, major browsers had to revoke their trust in all certificates issued by DigiCert Sdn. Bhd. (Note: DigiCert Sdn. Bhd. is not affiliated with the U.S.-based corporation DigiCert, Inc.) [9]. The above breach events illustrate how easy it is for CAs to make operational and management errors. The consequences of CA breaches can be quite severe. PKI security issues are also crucial for PGP as the present exchange among key servers and end-customers is conducted with SSL protocol using PKI certificates.

E. PGP key servers

PGP key servers are important component in OpenPGP system. Participants of PGP environment are supposed to upload their certificate updates including signatures of other participants, expiration dates, revocations, etc. If the sender does not have public key of the receiver on its local computer he may also download it from a key server.

However, if one wants to use PGP key servers infrastructure it is important to bear in mind the following:

- ***PGP key servers now can become more trustworthy***: based on security section in OpenPGP key server protocol[4], PGP key servers should not be trusted and must be used for informational purposes only. Basically it means that key servers do not provide full user access control, which can be misused by an attacker to upload compromised certificate of other users to the key server's storage. Blockchain-based certificate storage grants write access only to certificate holder's Ethereum account specified in the certificate (and also to administrator).
- ***Man-in-the-middle risk***: accessing PGP key servers is conducted using HTTP Key Server Protocol (HKP) which can be protected with SSL/TLS protocol. Given the potential vulnerabilities of PKI stemming from its single point of failure at CAs, it can be concluded that the whole PGP ecosystem becomes dependent on security of PKI. Indeed, if attacker can obtain a fake X.509 certificate with one of compromised CAs, as it was mentioned above, it can intercept the public key retrieve request and send a fake public key to the PGP participant. Blockchain fully resolves Man-in-the-Middle risk as the required certificate can be retrieved from the user's local replica of blockchain which cannot be altered due to hash reference check for each block.
- ***Key server synchronization delays***: PGP key servers provide important functionality like publication of certificate revocations which is not fully available in PKI environment. However the delays with PGP key server synchronizations which may take 15-30 hours can result in security breaches with compromised certificates already revoked by their owners.
- ***Key server should be active***: According to studies [10] there are more than 100 PGP key servers but some of them do not synchronize very often (up to several days and even more) implying risk of downloading revoked (compromised) certificates.

F. Outline

The rest of the paper is organized as follows: Section II provides a background of the PGP certificate validation mechanism, an introduction to the blockchain technology and how it can be used to build secure key server ecosystem. Section III explores the works related to revoked certificate logging, blockchain-based certificate management, etc. In Section IV we provide some details of our blockchain-based solution. Finally, Section V concludes the paper.

II. Background

In this section, first we present the idea behind the certificate revocation and different key server systems in PGP and PKI, which are crucial infrastructure for secure certificate validation. Then we provide a brief introduction to the blockchain technology and how it can be used to build key server management framework.

[4]https://tools.ietf.org/html/draft-shaw-openpgp-hkp-00

A. Revocation of certificates and validity in PGP and PKI

Revocation is an important part in certificate management as any certificate can be potentially compromised and there should be efficient procedure in place to quickly suspend the certificate validity. There are significant differences between PGP and PKI in the approach to revocation.

In PGP revocation can be conducted by a "revocator", a PGP participant holding revocation key for a given certificate. The idea here can be connected with the risk of operating system breach for a PGP certificate holder as she/he may not be able to revoke the certificate himself. The certificate updates related to its revocation should be uploaded to PGP key server to stop downloading of this certificate for encryption of sensitive information.

In PKI the certificate can be revoked by "introducer", i.e. a CA that issued the certificate. The certificate is then supposed to be published in Certificate Revocation List (CRL) of this CA, but as there is no universal revocation list yet available, the check of certificate validity in PKI is not straightforward. Based on X.509 concept [11], certificate validation should be conducted along "Chain of Trust", the path from a given certificate up to its issuing CA, then to parent CA and climbing the hierarchy up to the Root CA certificate (along the CA tree from the leaf to the root). All certificates along the chain of trust should be checked for validity.

B. Existing solutions to the certificate revocation security challenges

As we discussed above key servers play important role in supporting of OpenPGP authentication validation. According to a recent study the update of PGP total key storage is quite marginal and comes to 0.02 percent per day (1000 updates a day) [12]. According to the author, the efficiency of existing synchronization algorithms can be increased for that relatively minor updates as key server software was developed at time when the PGP key storage was smaller in terms of the number of uploaded certificates. Presently it takes 15-30 hours for certificate updates to be synchronized over the network of PGP key servers. The reduction of synchronization latency should reduce risks of using compromised public keys which were already revoked but not yet published with their new status on all key servers.

Traditional revoked certificate publication approaches used in PKI like CRL lack efficiency due to delays in publication of updated lists and absence of universal access point to all CRLs. However, recent initiatives like public PKI logs provide some solutions to existing certificate revocation challenges.

Log-based PKI approach has been proposed as a technique for fast publication of revoked certificates due to the problems stemming from CA breaches and compromised private keys. The idea behind is using highly-available public log servers that monitor and publish the certificates issued by CAs. These public logs provide transparency by ensuring that only publicly-logged certificates are accepted and trusted by end-customers, hence, any CAs' misbehavior will be detected. Google's Certificate Transparency (CT) [13] is the most widely deployed log-based PKI, and it is currently available in both Chrome and Firefox. In parallel, many proposals intend to extend the features of log-based PKI schema with further support for revocation and error handling. Unfortunately, despite these benefits, log-based PKIs still have several challenges related to the deployment process as explained in [14], [15].

Recently, many studies [16], [17], [18] propose implementation of blockchain technology to build secure PKI systems. Moreover some PKI blockchain-based management frameworks were already deployed and tested on Ethereum testnet [19]. Ethereum smart contracts proved to show good performance for X.509 certificate parsing and validation alongside extensive Chains of Trusts (up to 1200 certificates in a row). Parsing and verification of 1200 certificates took around 8 seconds on ordinary modern notebook. We will discuss blockchain solution to public key management challenges in details below.

C. Blockchain

Blockchain turns out to be one of the most intriguing technologies in the Internet industry today mainly due to success of Ethereum and other smart contract platforms like Hyperledger. The common key characteristics of all blockchain platforms are decentralization, persistency and auditability. Blockchain, also named distributed ledger, is an append-only data structure that stores transactions grouped into blocks to form the hash-chain of blocks. Each block references to the previous (parent) one as it contains the hash of the previous block. The first block of a blockchain is called *Genesis* block which has no parent/preceding block.

Currently most blockchains are used in financial industry, however more and more new implementations from different fields start to appear. Applications that require high reliability and full elimination of data manipulation risks can use blockchain. In addition, blockchain is distributed and can avoid the single point of failure situation. In recent years blockchains evolved to allow the execution of arbitrary logic known as *Smart Contracts*. Conceptually the smart contract is an application which runs on the top of blockchain and uses the underlying ordering of transactions to keep consistency of smart contract results between blockchain participants [20]. A smart contract code is executed by a network of *blockchain nodes* that reach consensus on the outcome of the execution, and update the contract's state storage on the blockchain.

In our approach, we used a separate instance of Ethereum blockchain which can support any currently available consensus (Proof-of-Work or Proof-of-Stake). By separate instance we mean set of isolated nodes using blockchain only for the purpose of our application. We believe it is better to use a separate blockchain instance for key storage infrastructure due to the following reasons:

- More computation-light consensus algorithms can be used, like Proof-of-Authority instead of Proof-of-Work. This leads to higher potential scalability in terms of transactions per seconds. Proof-of-Authority consensus implies permissioned nature of our blockchain implementation as only selected blockchain nodes (key servers) can confirm blockchain transactions.
- Much lower blockchain download time due to limited blockchain size. Basically we will have only PGP certificates in the blockchain.

Notably in the context of PGP key servers blockchain provides valuable security features such as write access based on Ethereum user accounts. Based on our implementation only administrator and the user corresponding to Ethereum account recorded in the certificate can update certificate data in the blockchain. Moreover, blockchain-based public key infrastructure, as a public append-only log, naturally provides the CT property proposed by Google [17].

III. Related work

Blockchain implementations for certificate services or digital identity systems were scrutinized by a number of different prior works. With exception of Sovrin/Indy embracing as many kinds of authentication as possible on one platform, all these works were focused on different aspects of PKI or, like Emercoin, just on safe storage and transfer of certificates eliminating Man-in-the-Middle risks.

One of the most interesting ideas in blockchain-based authentication appears to be Sovrin, which in 2017 became known as Hyperledger Indy. Sovrin's concept is based on DID, i.e. distributed ID, the identifier of a certain entity, usually corresponding to a private key [21]. In PGP context such DID can be considered as certificate's fingerprint, effectively SHA-1 hash of public key's core data.

Hyperledger Indy is declared as a public permissioned blockchain and appears to be one of few public blockchains in Hyperledger ecosystem. The concept of permissioned blockchain stems from Indy's idea of users' digital identity authentication by different entities including authorized state and public services, etc. The user can forward part of her/his digital identity (for instance driving license only, but not the passport) to some interested third parties. Although Indy/Sovrin declares itself as Web-of-Trust system, in reality it is likely to move towards traditional hierarchical PKI as more state agents are involved. Nevertheless if users could authenticate some "parts" of each other's digital identity in line with PGP concept in the final production version Indy can be scrutinized as a potential blockchain platform in the framework of this research in the future.

Apart from overwhelming concept of Indy/Sovrin there are some concrete implementation of blockchain-based certificate management. For instance, PKI management framework [19] was deployed on Ethereum's testnet. Golang-based REST service provides whole range of necessary functionality from issuing and revocation of X.509 certificates (including CA certificates) to smart contract-based verification along a whole Chain of Trust up to the root certificate.

Our experiments find that smart contract-based parsing and verification of long Chain of Trust consisting of 1200 certificates took around 8 seconds, while the verification of this Chain of Trust by Golang code retrieving the certificates from Ethereum took 15 sec, or twice slower compared to the smart contracts' performance. Each CA has a separate smart contract holding hashes of all issued certificates, revocation list and the certificate of this CA.

To link blockchain infrastructure to certificates the authors proposed hybrid X.509 certificates. Extensions of the hybrid certificates envisaged by X.509v3 standard contain issuing CA's smart contract address and, in case of CA certificate, the smart contract address of this CA.

Implementation of blockchain-based PKI was also announced by Emercoin in its EMCSSH project. Emercoin is a public blockchain quite close to Bitcoin in terms of architecture featuring the hybrid Proof-of-Work and Proof-of-Authority consensus depending on availability of mining capacity. Emercoin does not have smart contracts and stores the certificate hashes into blockchain. This means that the verification of the certificates is not distributed depending exclusively on the code outside blockchain.

Emercoin's EMCSSH is not focusing on Chain of Trust as by default the certificate does not contain links to its parent CA in the extension fields. On the contrary EMCSSH with just certificate hashes in its blockchain only mitigates the Man-in-the-Middle risk and makes administrators' life easier according to its creators.

Alternatively, Fromknecht et al. [18] leverage Certcoin to implement a blockchain-based PKI, storing domains and their associated public keys. Meanwhile in [17] authors scrutinize the privacy issue of the Certcoin. In [22] the authors propose Blockstack that uses Bitcoin blockchain to provide name registration system where the names are bound with public keys. In this work we mitigate the privacy issue since we will use the standard PGP certificate to develop our framework.

IV. Blockchain Implementation of PGP Key Servers

This section presents the details of our implementations of blockchain-based PGP key servers. Firstly we will provide an overview of the main benefits of the proposed approach. Then, we explain advantages of deployment of a separate instance of Ethereum blockchain rather than using traditional public Ethereum. After that we justify our approach of incorporating blockchain related information to PGP certificate. In the end of this section we will briefly describe the main functionality of the smart contract and its user interface.

A. Benefits of blockchain

Our blockchain implementation of key server resolves the challenges we mentioned in the section "PGP key servers" of Introduction. With Ethereum-based user account control the certificates can be uploaded or modified on a key server only from Ethereum account specified in the certificate. Blockchain technology also inherently resolves Man-in-the-Middle risk as all interested participants can have their own synchronized replica of blockchain which cannot be altered due to hash references to the previous block. Blockchain synchronization latency is usually capped by time period of new block formation varying from 15 seconds for Ethereum to 10 minutes for Bitcoin. Present synchronization latency of PGP key servers varies from several hours up to around one day.

B. Separate instance of blockchain

Another important consideration regarding blockchain could be the use of Ethereum separate instance (independent set of peers) rather than traditional public Ethereum blockchain. The benefits of Ethereum separate instance deployment are the following:

- Flexibility in blockchain consensus which can allow to substantially increase the performance. We propose to use permissioned blockchain with Proof-of-Authority (PoA) consensus, where transactions are approved by authorized nodes, rather than blockchain with Proof-of-Work (PoW) consensus implying heavy hash calculations conducted by miners.
- Potential difficulties with public Ethereum's blockchain downloading can be easily resolved with deployment of Ethereum separate instance. On February 2018 the size of Ethereum's public blockchain varied from 70GB to 350GB depending on state history storage options. Such blockchain size can be explained by massive smart contract deployments. As we interested only in PGP keys information stored in our smart contracts it is better to avoid dealing with all data in public Ethereum. In case of Ethereum separate instance the blockchain will only hold the data related to our application.
- Most importantly, participants of PGP infrastructure will not pay fees for loading data to blockchain. In case of traditional public blockchain transaction fees are paid to miners.

Present PGP key servers currently amounting to more than 100 nodes [10] are supported by the community. As PoA which can be used as consensus mechanism in PGP key server blockchain implies no substantial electricity expenses, the community's costs to support key server infrastructure are unlikely to increase with implementation of blockchain technology.

C. Design Methodology: incorporating Ethereum user address to PGP certificate

We built our application concept on the same principles implemented in [19] with hybrid X.509 certificates. Once blockchain information (in our case - the address of Ethereum user account) is integrated to certificate, we can link blockchain access control to user identity and design the whole security concept around it. Instead of expanding certificate data fields using X.509 extensions we decided to use Comment field of PGP user identity. Although to the best

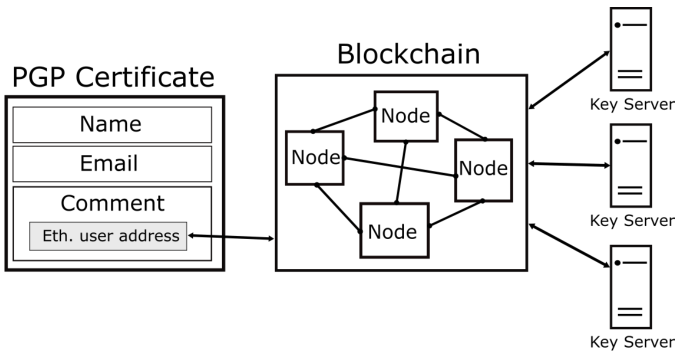

Fig. 1: Integrating blockchain user address in PGP certificate

of our knowledge restrictions on use of Comment field in PGP certificates were not published in the official documentation or in the research papers, many IT security specialists recommend to leave Comment field blank [5]. Indeed the Comment field is a part of PGP user identity along with name and e-mail address. When a user writes “Work“ or “I like strawberries“ in the Comment field of the certificate, the introducer also should confirm by signing this certificate that the certificate holder likes strawberries or will use this certificate at work. On the other hand for our purposes Comment field is a perfect fit. Introducer should confirm not only the name and email of certificate holder, but also her/his user address in Ethereum network corresponding to PGP key storage. As shown in Figure1 we use the Comment field in the following format: “blockchain:0x...“, thus Ethereum user address in hexadecimal format in lower case starting with “0x“ is written after keyword “blockchain:“

As we mentioned above the main concept of key server blockchain is based on write access restriction assuming that only administrator and a user with Ethereum address specified in the certificate’s Comment field can manipulate data related to the certificate. Administrator right control is realized with function modifier *onlyOwner* specified in the solidity contract *owned*, inherited by key server smart contract discussed below.

D. Implementation

The proposed blockchain-based PGP key server Proof-of-Concept includes two main parts: Unix application providing simple user interface to the key server and Ethereum smart contract providing key server core functionality.

1) Unix application: Command line application was developed with Golang IPC interface to Ethereum geth client assuming no Man-in-the-Middle risks which can be associated with RPC or REST interactions with Ethereum. Application extracts user data from local PGP certificates using GNU gpg client, parses certificate’s blockchain user account from Comment field and connects to Ethereum under this user account. Importantly, Ethereum private key corresponding to this user account should be locally stored in corresponding Ethereum folder of key server blockchain instance. Before each Ethereum connection user is asked to enter a password for Ethereum user.

[5]https://debian-administration.org/users/dkg/weblog/97

2) Smart contract: In contrast to PKI blockchain implementation [19] where each CA has its own smart contract we decided to implement a single smart contract to store all PGP certificates’ data.

The smart contract provides the following core functions:

- *checkRights*: validates the rights of the user address in the second parameter to change the blockchain data of PGP certificate identified by its fingerprint. Usually user address parameter is the current Ethereum user specified by built-in variable *msg.sender*
- *newCertificate*: uploads PGP certificate to blockchain alongside with all user data including her/his blockchain address. Rights of the user to upload the certificate are verified with *checkRights*. Event *evNewCertificateReturn* is emitted for performance control and error checks.
- *newSignt*: signs (introduces) certificate of another user and uploads the signed certificate to specifically designed storage of proposed certificates, not accessible by other participants. Only the holder of the signed certificate can download the certificate with *getProposedCert* function and/or publish it with *acceptProposedCert* function. Along with event *evNewSigntReturn* used for error control the event *evProposeCertSignature* is emitted to acknowledge the certificate holder regarding the signature.
- *revokeCert*: revokes PGP certificate with the user right validation using *checkRights* function. Notably, the user can perform the revocation only with the access to corresponding Ethereum account without use of revocation certificate. In our view this may be very convenient as it provides another protected way to revoke compromised certificates. Performance control and error checks are conducted with event *evRevokeCertificateReturn*
- *revokeSignt*: revokes user’s signatures (introductions) to PGP certificates of other participants. In OpenPGP concept it is impossible to revoke signatures, but we decided to include it into PoC as an experiment. The user right validation is performed with *checkRights* function. Event *evRevokeSigntReturn* is emitted for performance control and error check.
- *acceptPoposedCert*: as discussed above on certificate holder request copies signed certificate from introducer’s proposed certificate storage to certificate holder’s *ownCert* field. Certificate holder is authenticated with *checkRights* function. Event *evAcceptedCertSignature* is used for performance control and error check.

The important implications from the functionality above are the following:

- ***Full history of key server data is available*** due to Ethereum built-in API. Indeed it is possible to obtain the results of any read-only function (*view* or *pure* based on Solidity’s terminology) at any block in the past. The number of the block when some specific data was changed can be retrieved based on events we emitted within all core functions of our smart contract.

- ***Introducer who signed a certificate cannot upload it to key server without the certificate holder's acceptance*** which is in line with PGP best practices but implies additional restrictions on existing PGP key server functionality. Presently introducer can upload the signed certificate to PGP key server preferably after receiving consent from the certificate holder.
- ***Smart contract's versioning and code updatability should be handled***, as in contrast to [19] we have a single smart contract for all certificate data. There are a number of techniques helping to solve smart contract's code updatability without affecting its data in Ethereum. For instance, some advantages of Solidity's new opcode *delegatecall* can be exploited in our future work.

The BlockPGP software including Unix application and the smart contract is available as open source at https://github.com/alyakubov/blockpgp with detailed description of command line options and installation procedure.

V. Conclusions and future work

Key servers are the important part of OpenPGP Web-of-Trust infrastructure. Blockchain-based PGP key server Proof-of-Concept developed in this work solves a number of key servers' challenges and improves their performance. This is achieved by incorporating of blockchain-related data to PGP certificates. First, the developed Proof-of-Concept provides the write access rights to key server's PGP certificates only to the certificate holder (and to administrator), which reduces the risk of downloading compromised certificate from the key server. Second, blockchain resolves Man-in-the-Middle risk for key servers. Third, blockchain accelerates the synchronization between key servers to several minutes from several hours. Moreover, blockchain provides full history of a key server's states based on its built-in functionality.

Notably if we use a separate instance of blockchain (independent nodes dedicated from blockchain view point exclusively to key server tasks) the calculation-light consensus mechanisms can be used. This means that the total costs associated with the PGP infrastructure support by the community are unlikely to grow.

As future work, we plan to provide synchronization of our solution among existing key servers. Basically it means parallel storage of blockchain-protected certificates with incorporated blockchain user information and traditional certificates that are presently stored in PGP key servers. In this case the updates of blockchain-protected certificates in traditional PGP key servers must be scanned and resolved. We also plan to developed smart contract-based parsing and verification of PGP certificates.

References

[1] C. Adams and S. Lloyd, *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Professional, 2003.

[2] G. Caronni, "Walking the web of trust," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.(WET ICE 2000). Proeedings. IEEE 9th International Workshops on*. IEEE, 2000, pp. 153–158.

[3] D. Poddebniak, C. Dresen, J. Mller, F. Ising, S. Schinzel, S. Friedberger, J. Somorovsky, and J. Schwenk, "Efail: Breaking s/mime and openpgp email encryption using exfiltration channels," 2018.

[4] "The gnu openpgp privacy handbook," 1999.

[5] J. Yu and M. Ryan, "Evaluating web pkis," in *Software Architecture for Big Data and the Cloud*. Elsevier, 2017, pp. 105–126.

[6] D. Cooper, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," 2008.

[7] "Types of ssl certificates choose the right one," 2014.

[8] H. Anada, J. Kawamoto, J. Weng, and K. Sakurai, "Identity-embedding method for decentralized public-key infrastructure," in *International Conference on Trusted Systems*. Springer, 2014, pp. 1–14.

[9] J. Prins and B. U. Cybercrime, "Diginotar certificate authority breach'operation black tulip'," 2011.

[10] A. Ulrich, R. Holz, P. Hauck, and G. Carle, "Investigating the openpgp web of trust," in *ESORICS 2011: Computer Security*, 2011, pp. 489–507.

[11] "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, rfc 3280," 2002.

[12] A. Rucker, "An efficient pgp keyserver without prior context," 2017.

[13] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," Tech. Rep., 2013.

[14] S. Matsumoto and R. M. Reischuk, "Ikp: Turning a pki around with blockchains." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1018, 2016.

[15] S. Matsumoto, P. Szalachowski, and A. Perrig, "Deployment challenges in log-based pki enhancements," in *Proceedings of the Eighth European Workshop on System Security*. ACM, 2015, p. 1.

[16] K. Lewison and F. Corella, "Backing rich credentials with a blockchain pki," 2016.

[17] L. Axon and M. Goldsmith, "PB-PKI: A privacy-aware blockchain-based PKI," in *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017.*, 2017, pp. 311–318. [Online]. Available: https://doi.org/10.5220/0006419203110318

[18] C. Fromknecht, D. Velicanu, and S. Yakoubov, "Certcoin: A namecoin based decentralized authentication system," *Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep*, vol. 6, 2014.

[19] A. Yakubov, W. Shbair, A. Wallbom, D. Sandra, and R. State, "A blockchain-based pki management framework," in *IEEE/IFIP Man2block Conference Proceedings*, 2018.

[20] E. Androulaki, C. Cachin, A. D. Caro, A. Sorniotti, and M. Vukolic, "Permissioned blockchains and hyperledger fabric," *ERCIM News*, vol. 2017, no. 110, 2017. [Online]. Available: https://ercim-news.ercim.eu/en110/special/permissioned-blockchains-and-hyperledger-fabric

[21] S. Foundation, "Sovrin tm: A protocol and token for self-sovereign identity and decentralized trust. a white paper," 2018.

[22] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains." in *USENIX Annual Technical Conference*, 2016, pp. 181–194.