Secure digital certificates with a blockchain protocol

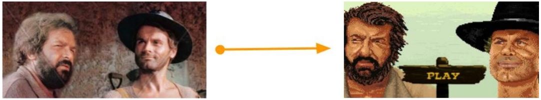
Federico Pintore[1]

Trento, 10^{th} February 2017

[1]University of Trento

Digital identities

In the digital world real people are identified with digital identities.



In a PKI (Public Key Infrastructure), to every digital identy corresponds a pair of cryptographic keys:

- the PUBLIC KEY, which is publicly known;
- the PRIVATE KEY, which must be kept secret.

Digital certificates

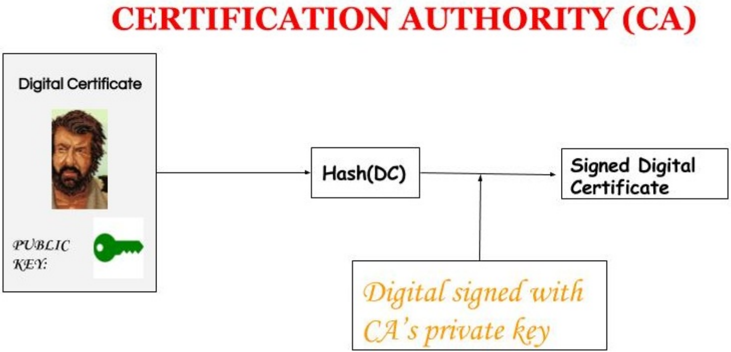
Digital identities are bound with corresponding public keys through **digital certificates**.

PKI

A PKI is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates.

X.509

X.509 is a widespread standard to manage digital certificates.



Mutually untrusted parties communicate through a **centralized system**.

Blockchain technology

The **Blockchain technology** is suitable for decentralized systems with mutually untrusted parties.

In 2015 Prof. Sead Muftic (KTH) proposed a blockchain-based protocol that allows distribution and management of digital certificates (linking a subject with his public key) without the need of CAs.

Muftic, Sead. "*Bix certificates: Cryptographic tokens for anonymous transactions based on certificates public ledger.* Ledger 1 (2016): 19-37.

BIX Certification Infrastructure (BCI)

1. New users **register** themselves to the system via an Instant Messaging (IM) system, named *BCI Instant Messaging System*;

BIX Certification Infrastructure (BCI)

1. New users **register** themselves to the system via an Instant Messaging (IM) system, named *BCI Instant Messaging System*;

2. during the registration, a new user receives a **unique identifier**, called *BIX Identifier*;

BIX Certification Infrastructure (BCI)

1. New users **register** themselves to the system via an Instant Messaging (IM) system, named *BCI Instant Messaging System*;
2. during the registration, a new user receives a **unique identifier**, called *BIX Identifier*;
3. after the registration, a new user interacts with the system via a *BCI Agent*, a **PC or smartphone application**.

Chains of certificates

Muftic's system is composed by **different blockchains of BIX certificates**, named *BCL's* (Bix Certificates Ledger).

Chains of certificates

Muftic's system is composed by **different blockchains of BIX certificates**, named *BCL's* (Bix Certificates Ledger).

Every *BCL* is a **chain of digital certificates**, cryptographically double-linked.

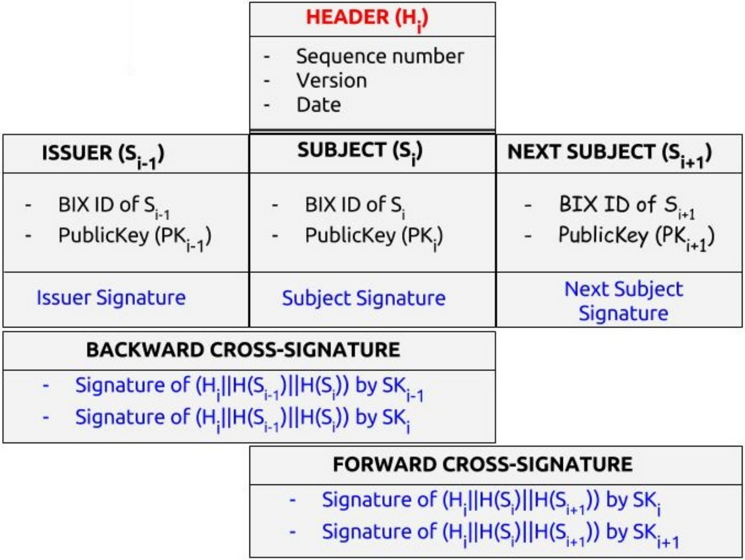
Chains of certificates

Muftic's system is composed by **different blockchains of BIX certificates**, named *BCL's* (Bix Certificates Ledger).

Every *BCL* is a **chain of digital certificates**, cryptographically double-linked.

After the registration, a new user can request the **issuing of a BIX certificate**, to be added to a preexisting *BCL* or to a new one.

BIX Certificates



Header (H_i)

- **Sequence number**: certificate's identification number, i, of the certificate (position in the BCL);
- **Version**: code designating the type of the BIX certificate;
- **Date/time**: date and time of issuance of the certificate.

Issuer, Subject and Next-Subject

Subject (S_i)

- **Subject BIX ID**: BIX Identifier of the user who owns the certificate;
- **Date/time**: date and time of creation of user's public/private key pair;
- **Algorithm identifier**: kind of asymmetric cryptographic scheme;
- **Public key**: cryptographic public key, PK_i, of the owner of the certificate.

Subject signature: digital signature over the Subject field via the private key SK_i associated to PK_i.

The Issuer (S_{i-1}) and the Next-Subject (S_{i+1}) are the Subject of the previous and the next certificate in the BCL.

Backward and Forward cross-signatures

BIX certificates are bound together through the:

- **Backward cross-signature**: contains two signatures, created by
  1. the Issuer S_{i-1},
  2. the Subject S_i,

  over the concatenation of the Header H_i, the hash of the Issuer $H(S_{i-1})$ and the hash of the Subject $H(S_i)$.

Backward and Forward cross-signatures

BIX certificates are bound together through the:

- **Backward cross-signature**: contains two signatures, created by
  1. the Issuer S_{i-1},
  2. the Subject S_i,

  over the concatenation of the Header H_i, the hash of the Issuer $H(S_{i-1})$ and the hash of the Subject $H(S_i)$.

- **Forward cross-signature**: contains two signatures, created by
  1. the Subject S_i,
  2. the Next Subject S_{i+1},

  over the concatenation of the Header H_i, the hash of the Subject $H(S_i)$ and the hash of the Next Subject $H(S_{i+1})$.

Special Certificates

Root certificate

- first certificate of a specific *BCL*;
- same structure of a standard certificate, but the Issuer field and the Subject field contain the same data.

Special Certificates

Root certificate

- first certificate of a specific *BCL*;
- same structure of a standard certificate, but the Issuer field and the Subject field contain the same data.

Tail certificate

- last certificate of a specific *BCL*;
- same structure of a standard certificate, but some fields are not populated (next user is still unknown).

The user that owns the tail certificate will become the issuer for the next certificate.

Certificate request

1. A new registered user creates his private and public keys;

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;
3. the owner of the tail certificate processes this request:

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;
3. the owner of the tail certificate processes this request:
    1. she (partially) fills the Issuer field and the Backward Cross-Signature field of the received certificate;

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;
3. the owner of the tail certificate processes this request:
   1. she (partially) fills the Issuer field and the Backward Cross-Signature field of the received certificate;
   2. she updates her BIX certificate (partially) filling the Next Subject field and the Forward Cross-Signature field;

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;
3. the owner of the tail certificate processes this request:
    1. she (partially) fills the Issuer field and the Backward Cross-Signature field of the received certificate;
    2. she updates her BIX certificate (partially) filling the Next Subject field and the Forward Cross-Signature field;
    3. she sends three certificates (root certificate, her certificate and his certificate) to the new user through the system.

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;
3. the owner of the tail certificate processes this request:
    1. she (partially) fills the Issuer field and the Backward Cross-Signature field of the received certificate;
    2. she updates her BIX certificate (partially) filling the Next Subject field and the Forward Cross-Signature field;
    3. she sends three certificates (root certificate, her certificate and his certificate) to the new user through the system.
4. the new user receives three certificates, completes them and requests all the *BCL* to the system in order to check its integrity;

Certificate request

1. A new registered user creates his private and public keys;
2. he creates and signs the Subject field of his certificate and sends it (as a request) to the system;
3. the owner of the tail certificate processes this request:
    1. she (partially) fills the Issuer field and the Backward Cross-Signature field of the received certificate;
    2. she updates her BIX certificate (partially) filling the Next Subject field and the Forward Cross-Signature field;
    3. she sends three certificates (root certificate, her certificate and his certificate) to the new user through the system.
4. the new user receives three certificates, completes them and requests all the *BCL* to the system in order to check its integrity;
5. he broadcasts to the system the completed certificates.

Certificate exchange

When a user S_i wants to perform a secure communication/transaction with a second user S_j, he sends his certificate c_i to S_j and requests her certificate c_j.

User S_i **checks certificate** c_j in two steps:

1. he verifies the Subject signature, the Issuer signature and also the Backward Cross-Signature of her certificate c_j.
2. S_i verifies that c_j is in the BCL, which is available in his local storage or must be updated.

Attack scenario - 1

An attacker tries to attach its certificate to a preexisting *BCL* without interacting properly with the last user of the *BCL*.

Attack scenario - 2

An attacker tries to corrupt a *BCL* built upon a root certificate of a preexisting *BCL*, resulting in another *BCL* that may re-distribute as a proper one.

Security of cryptographic schemes

Cryptographic schemes base their security upon the computational difficulty of solving some well-known mathematical problems.

Example
The RSA scheme bases its security upon the difficult to efficiently factorize huge integers.

⚠

The fact that to break a cryptographic scheme is **necessary to solve** a well-known mathematical problem is tipically only an unproven assumption.

Formal proof of security of a protocol

Goal

Model the possible attacks on the protocol and prove that a succesfull breach implies the solution of a hard, well-known mathematical problem.

If the mathematical problem cannot be solved, a **contradiction is reached** and the protocol is secure.

Some protocol's parameters must be chosen in such a way the problem guaranteeing the security becomes almost impossible to be solved in reasonable time.

Formal proof of security of a protocol

Adversary (A)

He tries to break the protocol making queries to C

Challenger (C)

They run the algorithm of the protocol.

Queries: private keys, encryption of specific plaintexts, decryption of specific ciphertexts...

Adversary's target

Recover a key, forge a digital signature,...

Formal proof of security of a protocol

General path

1. an *Assumption* is made: there is no polynomial-time algorithm solving a mathematical problem P with non-negligible probability;
2. the problem to break the protocol is reduced to solve the problem P;
3. if A breaks the protocol, then he is able to solve with non-negligible probability p_1 the mathematical problem.
    - a simulator S is build;
    - given an istance of P, S runs a challenger C that interacts with A, simulating the protocol correctly with non-negligible probability p_2;
    - S solves P with non-negligible probability (usually $p_1 p_2$).

Hash functions

Messages to be signed, seen as binary strings, are compressed via hash functions.

Definition (Hash functions)

A hash function H is a function of the form:

$$\begin{array}{rccc} H: & \{0,1\}^* & \to & \{0,1\}^\ell \\ & m & \mapsto & H(m) \end{array}$$

The image $H(m)$ is called *digest*.

Cryptographic hash functions

A hash function H is said cryptographic if some security assumptions holds. For example:

Definition (Collision resistance for R)

A hash function H is collision resistance if, given $R \subset \{0,1\}^r$, there is no polynomial-time algorithm finding distinct $m_1, m_2 \in L$ such that $H(m_1) = H(m_2)$ with non-negligible probability.

Examples

- SHA256, with digests of 256 bits;

ECDSA

The **Elliptic Curve Digital Signature Algorithm** (ECDSA) is a Digital Signature Scheme, i.e. an assymetric cryptographic scheme for producing and verifying digital signatures.

It consists of three algorithms:

1. *Key Generation*
2. *Signing*
3. *Verifying*

Key Generation

$$KeyGen(\kappa) \rightarrow (SK, PK)$$

Given

- a **security parameter** κ

it generates:

- a **public key** PK, that is published,
- a **secret key** SK.

Signing

$$Sign(m, SK) \rightarrow s$$

Given

- a **message** m,
- the **secret key** SK

it computes

- a **digital signature** s of m.

Verifying

$$Ver(m, s, PK) \rightarrow r$$

Given

- a **message** m,
- a **signature** s,
- the **public key** PK

it outputs

- the **result** $r \in \{True, False\}$ that says whether or not s is a valid signature of m computed by the secret key corresponding to PK.

Assumption on ECDSA

Definition (Security of a Digital Signature Scheme)

A Digital Signature Scheme DSS is said **secure** if an adversary A, given a public key PK - corresponding to a secret key SK - and some digital signatures $s_i = Sign(m_i, SK)$, is not able to identify a message $m \neq m_i \ \forall i$ and compute s such that $Ver(m, s, PK) = True$ in polynomial-time complexity with non-negligible probability.

We assume that ECDSA is **secure** since it bases its security upon the **Elliptic Curve Discrete Logarithm Problem** (ECDLP).

⚠

Nowadays, there is not a polynomial-time algorithm for solving the ECDLP.

Theorem 1

Theorem (Longo, _ , Sala, Rinaldo - 2016)

Let A be an adversary that manages to succesfully perform the **first attack** *with probability ϵ, then a simulator S might be built that, with probability at least ϵ, either solves the Collision Problem for the hash function relatively to the set L of all possible Subject fields, or breaks the Digital Signature Scheme.*

Corollary 1

Corollary (Longo, _ , Sala, Rinaldo - 2016)

If the Digital Signature Scheme used in Muftic's protocol is secure and the hash function is collision resistant for the set L, where L is the set of all possible Header fields, then the BIX protocol is secure against the first attack.

Theorem 2

Theorem (Longo, _ , Sala, Rinaldo - 2016)

Let A be an adversary that manages to succesfully perform the **second attack** *with probability ϵ, then a simulator S might be built that with probability at least $\frac{\epsilon}{n-1}$ either solves the Collision Problem for the hash function relatively to the set L of all possible Subject fields, or breaks the Digital Signature Scheme, where n is the length of the BCL that S gives to A.*

Corollary 2

Corollary (Longo, _ , Sala, Rinaldo - 2016)

If the Digital Signature Scheme used in Mutftic's protocol is secure and the hash function is collision resistant for the set L of all possible Subject fields, then the BIX protocol is secure against the second attack.

Thanks for your attention!