2017 9th International Conference on Cyber Conflict
Defending the Core
H. Rõigas, R. Jakschis, L. Lindström, T. Minárik (Eds.)
2017 © NATO CCD COE Publications, Tallinn

On the Security of the Blockchain BIX Protocol and Certificates

Riccardo Longo
Department of Mathematics
University of Trento
Trento, Italy
riccardolongomath@gmail.com

Federico Pintore
Department of Mathematics
University of Trento
Trento, Italy
federico.pintore@unitn.it

Giancarlo Rinaldo
Department of Mathematics
University of Trento
Trento, Italy
giancarlo.rinaldo@unitn.it

Massimiliano Sala
Department of Mathematics
University of Trento
Trento, Italy
maxsalacodes@gmail.com

Abstract: In recent years certification authorities (CAs) have been the target of multiple attacks due to their sensitive role in internet security. In fact, with access to malicious certificates it is possible to mount effective large-scale man-in-the-middle attacks that may become very vicious, especially if the incident is not properly handled. Many attacks, such as the 2011 ones against DigiNotar and Comodo, also show strong hints of state sponsorship; thus, CAs have to be considered primary targets in a scenario of (possibly state-sponsored) large-scale cyber attacks. Therefore, there is a need for a PKI protocol which is more resilient and without single points of failure, such as the CAs. The BIX protocol is a blockchain-based protocol that allows distribution of certificates linking a subject with their public key, hence providing a service similar to that of a PKI but without the need for a CA. In this paper, we analyse the security of the BIX protocol in a formal way. First, we identify formal security assumptions which are well-suited to this protocol. Second, we present some attack scenarios against the BIX protocol. Third, we provide formal security proofs that these attacks are not feasible under our previously established assumptions.

Keywords: *PKI, security proof, blockchain*

1. INTRODUCTION

Blockchain is an emerging technology that is becoming widely adopted to solve a myriad of problems where the classic centralised approach can be substituted by decentralisation. Indeed, centralised computations, albeit efficient, are possible only if there is a trusted third party (TTP) that everybody trusts. Nowadays, this is sometimes felt as a limitation and a possible vulnerability.

The general idea behind blockchain technology is that blocks containing information are created by nodes in the network, and these blocks are both public and cryptographically linked, so that an attacker should be unable to modify them without the users noticing the tampering. Also, the information contained in any block comes from the users and any user signs their own information cryptographically. Some examples of blockchain applications can be found in [1], [2] and [3].

A very sensitive security aspect which is usually kept centralised is the issuing of digital certificates, which form the core of a public key infrastructure (PKI). A certificate contains at least a cryptographic public key and it is digitally signed by a TTP. An example is X.509 certificates [4], mostly containing RSA public keys, which are widely used in the Internet for establishing secure transactions (e.g., an e-payment with an e-commerce site like Amazon). Since every user of a PKI must trust the certification authority (CA), which acts as a TTP, the identity of a web site is checked by verifying the CA's signature via the CA's public key. In a scenario of (possibly state-sponsored) large-scale cyber attacks, CAs may become primary targets because of their strategic role in guaranteeing the authentication and security of most web resources. Unfortunately, their role becomes a liability if they are compromised in the attack, since it becomes impossible for the attacked infrastructure to distinguish fake servers from real ones. In recent years, there have been multiple attacks against CAs, one of the most notable being the one that brought DigiNotar to bankruptcy in 2011. In that case, an intrusion led to the issuing of multiple malicious certificates and poor handling of the crisis left users exposed (with evidence of exploitation of these certificates in Iran) for months, and almost crippled the Dutch PKI. Other attacks saw prominent CAs among the targets, such as Comodo, StartSSL and Verizon. For details about the listed attacks we refer to [5].

Therefore, there is the need for a PKI protocol which is more resilient to wide cyber attacks and which does not introduce *single points of failure*, such as the CAs. This is exactly the idea behind the so-called BIX certificates. The BIX protocol aims to distribute the role of the CAs while preserving the security features. Indeed, the BIX protocol is designed with a blockchain-like structure that provides integrity to data, showcasing the possibility of a distributed PKI. A certificate is a block in a blockchain and a valid user interacting properly with the protocol will be able to attach their certificate to the blockchain. The protocol works with very few assumptions on the underlying network, but the original paper by Sead Muftic [6] focuses on the innovative ideas and the technology behind them, leaving formal proofs of security as a stimulating open research problem. Fascinated by his approach, in our present paper we prove the security of the BIX protocol, while providing suitable formal models for threat scenarios.

We achieve this by giving a mathematical reduction of the attacks to the solution of some (well-known) hard cryptographic problems. First, we suppose that an attacker tries to attach their certificate to a pre-existing certificate chain *without* interacting properly with the protocol. This is equivalent to having a malicious user trying to forge a valid certificate for themself (or for an innocent user). The second attack scenario considers that an adversary tries to modify an existing chain of certificates, distributing it as a proper chain.

In Section 2 of this paper we first define the security assumptions on which the security relies, giving formal definitions of the cryptographic primitives (i.e., hash functions and digital signatures) that act as the building blocks of the protocol, highlighting their security features that will eventually guarantee the security of the whole construction. In other words, the first step is the statement of supposedly intractable problems related to these primitives, which will become the goal of the formal reduction.

In Section 3 we provide a sketch of Muftic's scheme, highlighting the characteristics that are instrumental in its security.

In Section 4 and Section 5, we first proceed to formalise the threat scenarios and their actors, stating their capabilities and goals, in order to build realistic models of the attacks, suitable for formal analysis. This translation of protocol and malicious interaction into a formal language allows the reduction of an effective attack against the protocol to the disruption of the security of well-studied and established cryptographic primitives, such as hash functions and digital signatures.

Finally, we draw our conclusions regarding BIX protocol's resiliency against large-scale cyber attacks.

2. PRELIMINARIES

A. Formal Proofs of Security

In cryptography, the security of a scheme usually relies on the difficulty of a particular mathematical problem. So, in a formal proof of security the goal is to model the possible attacks on the scheme and prove that a successful breach implies the solution of a hard, well-known mathematical problem. Some security parameters may be chosen in such a way that the problem guaranteeing the security becomes almost impossible to solve in a reasonable time, and thus the scheme becomes impenetrable. More formally, the scheme is supposed secure if an *Assumption* holds on the related mathematical problem. Generally, an *Assumption* is that there is no *polynomial-time* algorithm that solves a problem $\mathcal{P}$ with *non-negligible* probability. For example, see the problem in the following subsection on hash functions.

In a formal proof of security of a cryptographic scheme there are two parties involved: a *Challenger* $\mathcal{C}$ that runs the algorithms of the scheme and an Adversary $\mathcal{A}$ that tries to break the scheme making queries to $\mathcal{C}$. In a *query* to $\mathcal{C}$, depending on the security model, $\mathcal{A}$ may request private keys, the encryption of specific plaintexts, and so on. The goal of $\mathcal{A}$ also depends on the security model, for example it may be to recover a key, or to forge a digital signature.

Hence, the security proofs follow a general path. Suppose there is an *Adversary* $\mathcal{A}$ that breaks the scheme with non-negligible probability p_1. A Simulator $\mathcal{S}$ is built such that if $\mathcal{A}$ breaks the scheme then $\mathcal{S}$ solves $\mathcal{P}$. So, given an instance of $\mathcal{P}$, $\mathcal{S}$ runs a challenger $\mathcal{C}$ that interacts with $\mathcal{A}$, simulating the scheme correctly with non-negligible probability p_2. Thus $\mathcal{S}$ solves $\mathcal{P}$ with non-negligible probability, which is usually p_1p_2, contradicting the Assumption.

To summarise, a formal proof of security is a reduction from the problem attack the scheme to the problem solve $\mathcal{P}$. Typically, $\mathcal{P}$ is a well-studied problem, so the assumption on its insolvability is accepted by the academic community.

B. Hash Functions

Commonly, the messages to be signed are compressed in fixed-length binary strings via a cryptographic hash function. A hash function H can be idealised as a function whose set of inputs is the set of all possible binary strings, denoted by $(\mathbb{F}_2)^*$, while its set of possible outputs is the set of all binary strings of given length (called *digest*). Real-life hash functions have a *finite* input set, but it is so large that can be thought of as infinite.

Cryptographic hash functions can need several security assumptions, however for the goals of this paper the following definitions are sufficient.

Definition 1: Collision Problem for a Class of Inputs. Let $r \geq 1l$, $h:(\mathbb{F}_1)^* \rightarrow (\mathbb{F}_2)^r$ be a *hash function*, and $L \subseteq (\mathbb{F}_2)^l$ be a class of inputs. The *collision problem* for h and L consists in finding two inputs $m_1,m_2 \in L$, with $m_1 \neq m_2$, such that $h(m_1) = h(m_2)$.

Definition 2: Collision Resistance of Hash Functions (Assumption 1). Let h be a *hash function*. We say that h is *collision resistant* for a class of inputs L if there is no polynomial-time algorithm $\mathcal{B}(h, L) \rightarrow \{m_1,m_2\}$ that solves the *Collision Problem* for h and L with non-negligible probability. The complexity parameter is r.

C. Digital Signatures and ECDSA

With the name *Digital Signature Scheme*, we refer to any asymmetric cryptographic scheme for producing and verifying digital signatures, consisting of three algorithms:

- *Key Generation* - $\mathsf{KeyGen}(\kappa) \rightarrow (\mathsf{SK}, \mathsf{PK})$: given a security parameter κ generates a public key PK, that is published, and a secret key SK.
- *Signing* - $\mathsf{Sign}(m, \mathsf{SK}) \rightarrow s$: given a message m and the secret key SK, computes a digital signature s of m.
- *Verifying* - $\mathsf{Ver}(m, s, \mathsf{PK}) \rightarrow r$: given a message m, a signature s and the public key PK, it outputs the result $r \in \{\mathsf{True}, \mathsf{False}\}$ that says whether or not s is a valid signature of m computed by the secret key corresponding to PK.

We measure the security of a Digital Signature Scheme by the difficulty of forging a signature in the following scheme (which results in an existential forgery):

Definition 3: Digital Signature Security Game. Let $\mathcal{DSS}$ be a Digital Signature Scheme. Its security game, for an adversary $\mathcal{A}$, proceeds as follows:

1) *Setup*
   The challenger $\mathcal{C}$ runs the KeyGen algorithm, and gives to the adversary the public key PK.
2) *Query*
   The adversary issues signature queries for some messages m_i the challenger answers giving $s_i = \mathsf{Sign}(m_i, \mathsf{SK})$.
3) *Challenge*
   The adversary is able to identify a message $\neq m_i \; \forall i$, and tries to compute s such that $\mathsf{Ver}(m, s, \mathsf{PK}) = \mathsf{True}$. If $\mathcal{A}$ manages to do so, they win.

Definition 4: Security of a Digital Signature Scheme (Assumption 2). A Digital Signature Scheme $\mathcal{DSS}$ is said to be secure if there is no polynomial-time algorithm $\mathcal{A}$ (w.r.t.κ) that wins the *Digital Signature Security Game* with non-negligible probability.

Ideally, a Digital Signature Scheme is designed in such a way that forging a signature in the scheme is equivalent to solving a hard mathematical problem. Although this equivalence is usually assumed but not proved, we say that the Digital Signature Scheme is based on that mathematical problem. Several Digital Signatures Schemes (e.g. [7]), are based on the discrete logarithm problem (DLOG), although other approaches exist, see e.g. [8], [9]. Among them, the Elliptic Curve Digital Signature Algorithm (ECDSA), which uses elliptic curves, is widespread. We refer to [10] for the details about the design of ECDSA.

If an attacker is able to solve the DLOG on an elliptic curve $\mathbb{E}$, then they can break the corresponding ECDSA. The converse is much less obvious. In [12], the authors provide convincing evidence that the unforgeability of several discrete logarithm-based signatures cannot be equivalent to the DLOG problem in the standard model. Their impossibility proofs apply to many discrete logarithm-based signatures like DSA, ECDSA and KCDSA, as well as standard generalisations of these. However, their work does not explicitly lead to actual attacks. Assuming that breaking the DLOG is the most efficient attack on ECDSA, then nowadays recommended key lengths start from 160 bits.

3. A DESCRIPTION OF BIX CERTIFICATES

In this section, we describe the BIX certificates and the structure containing them, called the BIX Certification Ledger (BCL). BIX certificates share many similarities with X.509 certificates, but the identities are anonymous. For a detailed comparison, we refer to [6, Section 2.1]; here we only highlight their characteristics that are instrumental for our security proofs.

The BCL collects all the BIX certificates filling a double-linked list, in which every certificate is linked to the previous and the next. To simplify our notation, we define the BCL as a 'chain of certificates', CC, of n certificates, that we may consider as a sequence:

$$\mathrm{CC}: c_0, \ldots, c_{n-1}.$$

We denote with λ a function returning the length of a chain, that is $\lambda(\mathrm{CC}) = n$. Also, $||$ denotes string concatenation.

TABLE 1. STRUCTURE OF A BIX CERTIFICATE

	Header (H_i)	
	Sequence number Version, Date.	
Issuer (S_{i-1})	Subject (S_i)	Next Subject (S_{i+1})
Bix ID of S_{i-1} Public key (PK_{i-1})	Bix ID of S_i Public key (PK_i)	Bix ID of S_{i+1} Public key (PK_{i+1})
Issuer Signature	Subject Signature	Next Subject Signature
Backward cross-signature		
Signature of $(H_i \| \| h(S_{i-1}) \| \| h(S_i))$ by SK_{i-1} Signature of $(H_i \| \| h(S_{i-1}) \| \| h(S_i))$ by SK_i		
	Forward cross-signature	
	Signature of $(H_i \| \| h(S_i) \| \| h(S_{i+1}))$ by SK_i Signature of $(H_i \| \| h(S_i) \| \| h(S_{i+1}))$ by SK_{i+1}	

Remark 1. The owner of the certificate c_i has a double role: as a **user**, with c_i that certificates their identity; as an **issuer**, providing the certificate c_{i+1} to the next user. In this way, there is no need of a CA.

To each certificate corresponds a user having a pair private key/public key, which we denote with SK_i and PK_i. Certificate c_0 is called the root certificate and certificate c_{n-1} is called the tail certificate.

In this paper, a certificate c_i for $i = 1, \ldots, n-2$ is defined by the following fields (and subfields) (necessary for our proofs of security), while the complete list can be found in [6]. Root and tail certificates are described later on.

1) Header ($\boldsymbol{H}_i$ *)*

In this field, there is general information such as timestamps and protocol version, but the relevant information for our analysis is the **Sequence number** i, that is the identification number of the certificate and also the position with respect to certificates of other BIX members.

2) Subject ($\boldsymbol{S}_i$)
The Subject contains the personal information that identifies the i-th user (S_i), in particular:

- **Subject BIX ID.** The unique global identifier of the user who owns the certificate. All BIX IDs contained in the Subject fields of a valid chain are distinct.
- **Public key.** The cryptographic public key of the owner of the certificate PK_i.

3) Subject signature
This contains the signature over the Subject attributes via the private key SK_i associated to PK_i.

4) Issuer ($\boldsymbol{S}_{i-1}$)
The Issuer field enjoys the same attribute structure as the Subject field, but it identifies the BIX member who certified S_i, i.e., it contains the Subject attributes of c_{i-2}, which identifies S_{i-2} (the previous member in the BCI).

5) Issuer signature
This field contains the signature over the Issuer attributes created by the Issuer, that is, performed via the private key SK_{i-1} associated to PK_{i-1}.

6) Backward cross-signature
The Backward_Cross_Signature contains two signatures, one created by the Issuer S_{i-1} and the other created by the Subject S_i, over the same message: $(H_i||h(S_{i-1})||h(S_i))$. Note that this field guarantees validity of the Header and binding between the Subject and the Issuer.

7) Next Subject ($\boldsymbol{S}_{i+1}$)
The Next_Subject field enjoys the same attribute structure of the Subject field, but it identifies the BIX member who is certified by S_i, i.e., it contains the Subject attributes of c_{i+1}, which identifies S_{i+1} (the next member in the BCI).

8) Next Subject signature
This is the same field as Subject signature, except it is created by the Next Subject over its own data, that is, performed via the private key SK_{i+1} associated to PK_{i+1}.

9) Forward cross-signature
The Forward_Cross_Signature contains two signatures, one created by the Subject S_i and the other created by the Next Subject S_{i+1}, over the same message: $(H_i||h(S_i)||h(S_{i+1}))$.

Note that this field guarantees binding between the current user acting as an issuer and the next user (to whom the next certificate c_{i+1} is issued).

We now describe the special certificates:

- The certificate c_0, called the *root certificate*, has the same structure of a standard certificate, but the Issuer field and the Subject field contain the same data. Indeed, the root user S_0 is not a normal user but rather an entity that initiates the specific BCL.

- The certificate c_{n-1} has the same structure of a standard certificate, but some fields are not populated because the next user is still unknown: Next_Subject, the Next_Subject signature, the Forward_Cross_Signature. However, we underline that it is regularly published in the chain and considered valid by other users.
  The last user that owns the last certificate, c_{n-1} will then become the issuer for the next certificate (see *Remark 1*).

In the BIX protocol a new user requests the issuing of a new certificate through a query to the BIX community, which is processed only by the user that owns the tail certificate of the chain. For further details about the BIX protocol we refer to [6, Section 3.3].

4. CHAIN LENGTHENING ATTACK SCENARIO

The first attack scenario that we consider supposes that an attacker tries to attach their certificate to a pre-existing certificate chain without interacting properly with the last user of the chain. More precisely, the attacker $\mathcal{A}$ should not interact with the subject of the last certificate in the chain according to the BIX protocol.

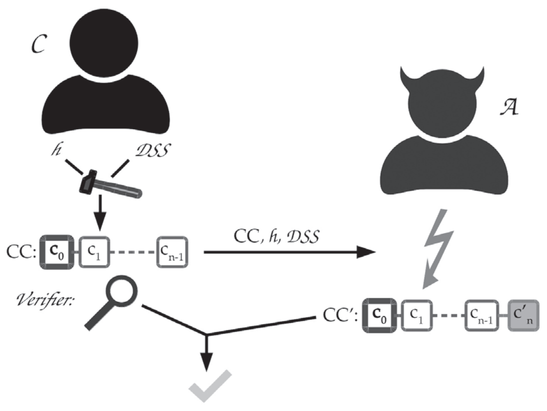
A. The Security Game

For this attack, we consider a game where an adversary $\mathcal{A}$ aims to add a certificate to the tail of a certificate chain CC. We will call it *Static Chain Lengthening (SCL) Game* and it proceeds as follows:

- The challenger $\mathcal{C}$ builds a certificate chain CC according to the BIX protocol with root certificate c_0, using a hash function h and a digital signature scheme $\mathcal{DSS}$.
- $\mathcal{C}$ passes to $\mathcal{A}$ the chain CC together with h and $\mathcal{DSS}$.
- $\mathcal{C}$ builds an honest verifier $\mathcal{V}$ that given a certificate c^* and a certificate chain CC* outputs True if the root certificate of CC* is c_0 and c^* is a valid certificate of CC*, False otherwise.
- $\mathcal{A}$ tries to build a forged certificate chain CC$'$, $\lambda(\text{CC}') = n + 1$, such that:
  - o CC$'$ truncated before the last certificate c'_n is identical to CC if the Next_Subject and Forward_Cross_Signature fields of the second-to-last certificate of CC$'$ are not considered (i.e. we obtain CC$'$ by adding a certificate to CC and completing c_{n-1} accordingly);
  - o user S_{n-1} did not take part in the creation of c'_n and so in particular they did not perform the Forward_Cross_Signature of c_{n-1} and the Backward_Cross_Signature of c'_n;
  - o $\mathcal{V}(c', \text{CC}') = \text{True}$ where c'_n is the last certificate of CC$'$.

$\mathcal{A}$ wins the SCL Game if they build a CC$'$ that satisfies these last three points.

FIGURE 1. SCL GAME



Definition 5: Security against SCL. The BIX protocol is said to be **secure against static chain lengthening** if there is no adversary $\mathcal{A}$ that in polynomial time wins the *SCL Game* with non-negligible probability.

B. Security Proof

In the following we will prove that winning the SCL game implies the contradiction of our security assumptions introduced in Section 2.

Theorem 1. Let $\mathcal{A}$ be an adversary that wins the *SCL Game* with probability ϵ. Then a simulator $\mathcal{S}$ might be built that, with probability at least ϵ, either solves the *Collision Problem*, with L the set of all possible Subject fields, or wins the *Digital Signature Security Game*.

Proof. Let $\mathcal{DSS}$ be the digital signature scheme and h the hash function used in the BIX protocol, and $L \subseteq (\mathbb{F}_2)^l$ be the class of all possible Subject fields. We will build a simulator $\mathcal{S}$ that simultaneously plays the *Digital Signature Security (DSS) Game* and tries to solve an instance of the *Collision Problem* for L. It does so by simulating an instance of the *SCL Game* and exploiting $\mathcal{A}$. We will prove that if wins the SCL Game then either $\mathcal{S}$ finds a solution for the Collision Problem or $\mathcal{S}$ wins the DSS Game.

$\mathcal{S}$ starts with taking as input an instance (h, L) of the Collision Problem and a public key PK^* given by the $\mathcal{DSS}$ challenger (i.e., the output of the first phase of the DSS Game for the scheme $\mathcal{DSS}$). $\mathcal{S}$ then proceeds to build a certificate chain CC^* following the BIX protocol. $\mathcal{S}$ builds all but the last certificate normally, running the KeyGen algorithm of the $\mathcal{DSS}$ to choose public keys for the Subject fields, so the corresponding secret keys are available to sign these certificates properly. Then let $n = \lambda(\mathrm{CC})^* \geq 2$ (i.e. the number of certificates contained in CC^*), c_0^* its root certificate and c_{n-1}^* the last one. $\mathcal{S}$ sets the Subject of c_{n-1}^*, that we will denote by

S^*_{n-1}, such that its public key is PK^*, then it queries the challenger of the DSS Game to obtain three valid signatures, respectively, on:

- the hash $h(S^*_{n-1})$ of this subject,
- $(H^*_{n-1}||h(S^*_{n-2})||h(S^*_{n-1}))$ for the Backward_Cross_Signature of c^*_{n-1},
- $(H^*_{n-2}||h(S^*_{n-2})||h(S^*_{n-1}))$ for the Forward_Cross_Signature of c^*_{n-2},

where H^*_{n-2} is the Header of c^*_{n-2}, H^*_{n-1} is the Header of c^*_{n-1}, and $h(S^*_{n-2})$ is the hash of the Issuer of c^*_{n-1}, that is the Subject of c^*_{n-2}. In this way $\mathcal{S}$ completes a certificate chain CC^* of length n, that it passes to $\mathcal{A}$.

$\mathcal{A}$ responds with a counterfeit chain CC' of length $\lambda(\mathrm{CC}) = n + 1$. If CC' is not valid (the chains CC' and CC^* do not correspond up to the n-th certificate, or an integrity check fails) then $\mathcal{S}$ discards this answer and gives up ($\mathcal{S}$ fails).

Otherwise, if the verifier outputs True, the chain CC' is valid. Denote by l' the string $(H_n'||h(S_{n-1}')||h(S_n'))$ signed in the Backward_Cross_Signature of c'_n (the last certificate of CC') by the private key corresponding to PK^*. We have two cases:

- l' is equal to a message for which $\mathcal{S}$ requested a signature.
  Because of its bit-length, l' may be equal to $l^*_0:=(H^*_{n-2}\,||h(S^*_{n-2})||h(S^*_{n-1}))$ or $l^*_1:=(H^*_{n-1}\,||h(S^*_{n-2})||h(S^*_{n-1}))$, but not to $h(S^*_{n-1})$. In either case, $l'=l^*_0$ or $l'=l^*_1$, the equality implies that $h(S_n')=h(S^*_{n-1})$, but the specification of the BIX protocols supposes that different certificates have a different BIX ID in the Subject (and we know that CC' is valid). So $S'_{n-1} = S^*_{n-1} \neq S'_n$, because of the BIX ID's, but they have the same hash so $\mathcal{S}$ may submit (S^*_{n-1}, S'_n) as a solution to the Collision Problem.
- l' is different from all messages for which $\mathcal{S}$ requested a signature.
  In the Backward_Cross_Signature of c'_n there is a signature s of l' such that $\mathsf{Ver}(l', s, \mathsf{PK}*) = \mathsf{True}$ (remember that PK^* is the public key of the Issuer of c'_n and that CC' is considered valid, so the signatures check out), so $\mathcal{S}$ may submit (l', s) as a winning answer of the challenge phase of the DSS Game.

So, if $\mathcal{S}$ does not fail, it correctly solves the Collision Problem or wins the DSS Game, and since $\mathcal{A}$ is a polynomial-time algorithm, $\mathcal{S}$ is a polynomial-time algorithm too, given that the other operations performed correspond to the building of a certificate chain and this must be efficient. $\mathcal{S}$ might fail only if the chain given by $\mathcal{A}$ is not valid (i.e. if $\mathcal{A}$ fails). Since the simulation of the SCL Game is always correct, $\mathcal{A}$'s failure happens with probability $1 - \epsilon$, then the probability that $\mathcal{S}$ wins is $1 - (1 - \epsilon) = \epsilon$.

Corollary 1: *SCL Security*. If the DSS is secure *(Assumption 1)* and the hash function is collision resistant for the class L *(Assumption 2)*, where L is the set of all possible Subject fields, then the BIX protocol is secure against the Static Chain Lengthening.

Proof. Thanks to *Theorem 1*, given a polynomial-time adversary that wins the SCL Game with non-negligible probability ϵ, a polynomial-time simulator might be built that with the same probability either solves the *Collision Problem* or wins the *DSS Game*. So, let C be the event 'solution of the Collision Problem' and D be the event 'victory at the DSS Game'. We have that

$$\epsilon = P(C \vee D) \leq P(C) + P(D)$$

The sum of two negligible quantities is itself negligible, so the fact that ϵ is non-negligible implies that at least one of $P(C)$ and $P(D)$ is non-negligible, and this means that *Assumption 1* or *Assumption 2* is broken.

Remark 2. The infeasibility of the above attack guarantees also the non-repudiation property of the last certificate in the chain. That is, if Alice (the user of the second-to-last certificate) tries to repudiate Bob (the user of the last certificate), with an eye to issuing another certificate, then Bob might claim his rightful place showing a version of the chain containing his certificate. This chain is then the proper one, since no one can attach its certificate to the tail of the certificate chain without being a proper user.

Remark 3. We have assumed (see *Assumption 2*) that $\mathcal{DSS}$ is secure against existential forgery, but in the proof of Theorem 1 the freedom of the attacker in the choice of the message to be signed is limited. In fact, it has to forge a signature of $l' := (H_n' || h(S_{n-1}') || h(S_n'))$, where $h(S_{n-1}')$ is given by $\mathcal{S}$, and even H_n' is not completely controlled by $\mathcal{A}$ (e.g. the sequence number is given). So, a large part of the string to be signed is beyond the control of the forger, hence the challenge is something in between an existential and a universal forgery, which is the weaker assumption on the Digital Signature Scheme ($\mathcal{DSS}$). However, the security of $\mathcal{DSS}$ against universal forgery is not sufficient for our purposes, so we settled on the stronger assumption.

5. CERTIFICATE TAMPERING

In the second attack scenario that we consider, a malicious attacker tries to corrupt a chain of certificates built on a trusted root certificate, resulting in another chain that may redistribute as a proper chain with the same root but with altered information. As we will see, the security against this attack would guarantee that no external attacker can modify any certificate in the chain, including deleting or inserting a certificate in any non-ending point, as long as the root certificate is safe (no unauthorised use), secure (cannot be broken) and public (anyone can check it). If the security proved in the previous section is also considered, then a certificate chain is also secure at the end point (no one can wrongfully insert themself at the end or disavow the last certificate) achieving full security from external attacks to the BIX protocol.

A. The Security Game

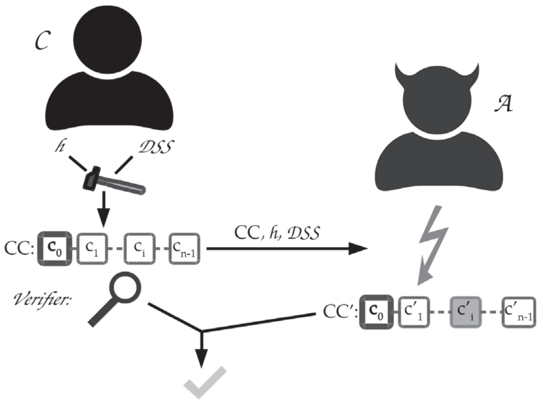
For this attack, we consider a game where an adversary $\mathcal{A}$ aims to modify information contained in the *Subject* field of a certificate c_i contained in a certificate chain $\mathtt{CC}$, with $1 \leq i \leq n-1$,

$n = \lambda(\mathrm{CC})$. We will call it the *Static Tampering with Subject (STS) Game* and it proceeds as follows:

- The challenger $\mathcal{C}$ builds a certificate chain CC with root certificate c_0, according to the BIX protocol and using a hash function h and a Digital Signature Scheme $\mathcal{DSS}$. Let $n = \lambda(\mathrm{CC})$.
- $\mathcal{C}$ passes to $\mathcal{A}$ the chain CC together with h and $\mathcal{DSS}$.
- $\mathcal{C}$ builds an honest verifier $\mathcal{V}$ that, given a certificate c^* and a certificate chain CC* outputs True if the root certificate of CC* is c_0 and c^* is a valid certificate of CC*, False otherwise.
- $\mathcal{A}$ tries to build a forged certificate chain CC$'$ such that:
  - exists $1 \leq i \leq n-1$ such that Subject fields of c_i and c'_i are different, that is, $S_i \neq S_i'$.
  - $\mathcal{V}(c_i', \mathrm{CC}') = \mathtt{True}$

$\mathcal{A}$ wins the *STS* Game if they successfully build such a CC$'$ satisfying the last two items.

FIGURE 2. STS GAME



Definition 6: Security against STS. The BIX protocol is said secure against Static Tampering with Subject if there is no adversary $\mathcal{A}$ that in polynomial time wins the *STS Game* with non-negligible probability.

B. Security Proof

In the following, we will prove that winning the SCL game implies the contradiction of our security assumptions introduced in Section 2.

Theorem 2. Let $\mathcal{A}$ be an adversary that wins the *STS Game* with probability ϵ. Then a simulator $\mathcal{S}$ might be built that with probability at least $\frac{\epsilon}{n-1}$ either solves the *Collision Problem*, where L is the set of all possible Subject fields, or wins the *Digital Signature Security Game*, where n is the length of the certificate chain that $\mathcal{S}$ gives to $\mathcal{A}$.

Proof. Let $\mathcal{DSS}$ be the Digital Signature Scheme and h the hash function used in the BIX protocol, and $L \subseteq (\mathbb{F}_2)^l$ be the class of all possible Subject fields. We will build a simulator $\mathcal{S}$ that simultaneously plays the digital signature security (DSS) Game and tries to solve an instance of the Collision Problem for L. It does so by simulating an instance of the STS Game and exploiting $\mathcal{A}$. We will prove that when $\mathcal{A}$ wins the STS Game, at least one in $n-1$ times $\mathcal{S}$ is successful. To be more precise, if $\mathcal{S}$ does not find a solution for the collision problem then $\mathcal{S}$ wins the DSS Game.

$\mathcal{S}$ starts with taking as input an instance (h, L) of the Collision Problem and a public key PK^* given by the $\mathcal{DSS}$ challenger (i.e., the output of the first phase of the Digital Signature Security Game for the scheme $\mathcal{DSS}$).

$\mathcal{S}$ now proceeds to build a certificate chain CC following the BIX protocol, as follows. First, $\mathcal{S}$ chooses $n \geq 2$ (possibly depending on the $\mathcal{A}$'s requirements). Then $\mathcal{S}$ selects $1 \leq k \leq n-1$ at random to be the index of a certificate c_k in CC. $\mathcal{S}$ builds the first $k-1$ certificates normally, running the KeyGen algorithm of the $\mathcal{DSS}$ scheme to choose public keys for the Subject fields, so the corresponding secret keys are available (to $\mathcal{S}$) to sign these certificates properly. So $c_0,\ldots, c_{k-3}$ are complete certificate and c_{k-2} is a tail certificate. Then it sets the Subject of c_{k-1} such that its public key is PK^*, and a header H_{k-1}. It queries the challenger of the DSS Game to obtain three valid signatures, respectively, on:

- the hash $h(S_{k-1})$ of this subject,
- $(H_{k-1}||h(S_{k-2})||h(S_{k-1}))$ for the Backward_Cross_Signature of c_{k-1} (if $k > 1$),
- $(H_{k-2}||h(S_{k-2})||h(S_{k-1}))$ for the Forward_Cross_Signature of c_{k-2} (if $k > 1$),

where we recall that H_{k-2} is the Header of c_{k-2}, H_{k-1} is the Header of c_{k-1}, $h(S_{k-2})$ is the hash of the Issuer of c_{k-1}. Then $\mathcal{S}$ builds the $k+1$-th certificate, choosing a H_k and S_k, using again the KeyGen algorithm to sign S_k, querying the DSS challenger for two valid signatures, respectively, on:

- $(H_k||h(S_{k-1})||h(S_k))$ for the Backward_Cross_Signature of c_k,
- $(H_{k-1}||h(S_{k-1})||h(S_k))$ for the Forward_Cross_Signature of c_{k-1},

where we recall that H_k is the Header of c_k and $h(S_k)$ is the hash of the Subject of c_k. Finally, $\mathcal{S}$ completes the chain CC (following the protocol and choosing everything, including the SK_i's), so that it has n certificates, and passes it to $\mathcal{A}$.

$\mathcal{A}$ responds with a counterfeit chain CC'. $\mathcal{A}$ fails if and only if CC' is not valid, which happens when there is no $1 \leq i \leq n-1$ such that $S'_i \neq S_i$ or when the integrity check of the verifier fails.

If we are in this situation, $\mathcal{S}$ discards CC' and gives up ($\mathcal{S}$ fails).

Otherwise, let $1 \leq i \leq n-1$ be the first index for which $S'_i \neq S_i$. Since k is chosen at random, we have that $k = i$ with probability $\frac{1}{n-1}$. In this case, there are two possibilities:

- $h(S_k) = h(S'_k)$, but $S_k \neq S_k'$ for hypothesis, then $\mathcal{S}$ outputs the pair (S_k, S'_k) as a solution to the collision problem.
- Otherwise, we have that $S_{k-1} = S_{k-1}'$ and PK^* is the public key of the issuer of c_k'. Then in the Backward_Cross_Signature of the certificate c_k' there is the digital signature s for which holds the relation $\mathrm{Ver}((H_k'||h(S_{k-1}')||h(S_k')), s, \mathrm{PK}^*) = \mathrm{True}$ (remember that CC' is considered valid, so the signatures check out). So $\mathcal{S}$ may submit $((H_k'||h(S_{k-1}')||h(S_k')), s)$
- as a winning answer of the challenge phase of the DSS Game, since it is different from the messages $\mathcal{S}$ queried for signatures, that are
  $[h(S_{k-1}), (H_{k-1}||h(S_{k-2})||h(S_{k-1})), (H_{k-2}||h(S_{k-2})||h(S_{k-1})),$
  $(H_k||h(S_{k-1})||h(S_k)), (H_{k-1}||h(S_{k-1})||h(S_k))]$

So $\mathcal{S}$ correctly solves the collision problem or wins the DSS Game at least when $\mathcal{A}$ wins and $i{=}k$. The probability of this event is at least the probability of the two cases and so it is

$$\epsilon \cdot \frac{1}{n-1} = \frac{\epsilon}{n-1}$$

Note also that since $\mathcal{A}$ is a polynomial time algorithm, so is $\mathcal{S}$.

Corollary 2: STS Security. If the DSS is secure (see *Assumption 2*) and the hash function is collision resistant for the class L (*Assumption 1*) where L is the set of all possible Subject fields, then BIX protocol is secure against the Static Tampering with Subject.

Proof. For the BIX protocol to be functional the length of the chain must be polynomial. So, for the result of *Theorem 2*, given a polynomial time adversary that wins the STS Game with non-negligible probability ϵ, a polynomial time simulator might be built that with probability at least $\frac{\epsilon}{n-1}$ either solves the Collision Problem or wins the DSS Game, where n is the length of the chain. So, let C be the event ‘solution of the Collision Problem’ and D be the event ‘victory at the DSS Game’. We have that

$$\frac{\epsilon}{n-1} \leq P(C \vee D) \leq P(C) + P(D)$$

The fact that $\frac{\epsilon}{n-1}$ is non-negligible implies that at least one of $P(C)$ and $P(D)$ is non-negligible, and this means that *Assumption 1* or *Assumption 2* is broken.

6. CONCLUSIONS

In this paper, the BIX certificates protocol proposed in [6] has been formally analysed from a security point of view. In particular, the security against static attacks that aim to corrupt a chain has been proven, reducing the security to the choice of adequate hash function and digital signature scheme. For this reason, the security of ECDSA, the main DSS nowadays, has also been discussed.

The current BIX protocol is still insufficiently complete for it to be considered a full PKI. Possibly, the main lack is the absence of a procedure to revoke or to renew certificates. This is an open problem and further research effort is needed. However, the security proofs given in this paper show how the BIX infrastructure is a reliable structure for storing public identities in a distributed and decentralised way. While a targeted attack on a CA can result in the issuing of malicious certificates or revocation of valid ones, shattering every certificate it issued, in the case of a cyber attack BIX certificates could still be trusted because no single entity could be targeted and exploited to take down the entire system. Indeed, we suppose that BIX certificates are issued and distributed in *peacetime*, so that when an emergency breaks out the infrastructure is ready to cope with possible attacks. Indeed, the properties proven in this work guarantee the integrity of the information contained in the certificate chain, so users can rely upon it even in the middle of a cyber attack. It is true that a targeted offensive against the owner of the last certificate would disrupt the protocol, preventing the issuing of new certificates. Nevertheless, if this user is taken down, the validity of existing certificates will still hold.

It may seem that the BIX protocol relies on a trusted third party, the messaging system. However, it is not a third party, as highlighted by Muftic [6], since it only passively broadcasts certificates and for purely addressing purposes it verifies the uniqueness of BIX identifiers (in Muftic's construction the IM protocol used is [13]).

Our conclusion is that a PKI system based on the BIX protocol is more resilient to a wide-scale cyber attack than the standard PKI protocols based on CAs.

Regarding related research, the idea of using a public ledger for digital identities has prominent applications in the distribution of Bitcoin wallet addresses (see for example [14]), but there are also applications that try to leverage the functionalities of cryptocurrencies to improve PKI. For example, Matsumoto and Reischuk [15] exploit smart contracts on Ethereum to deter misbehaviour of CAs and to incentive extended vigilance over CAs' behaviour. However, we fear that this is not sufficient in case of a large-scale cyber attack, because the financial losses that this solution enforces would affect the attacked CA and not the attackers themselves.

REFERENCES

[1] S. Wilkinson et al., *Storj: A Peer-to-Peer Cloud Storage Network*, 2014 [Online]. Available: http://storj.io/storj.pdf.

[2] M. Araoz, *Proof of Existence*, 2015 [Online]. Available: https://proofofexistence.com/about.

[3] Ethereum Foundation, *Ethereum Project*, 2015 [Online]. Available: https://www.ethereum.org/.

[4] D. Cooper et al., ‘Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile’, IETF RFC 5280, 2008.

[5] A. Niemann, and J. Brendel, *A survey on CA compromises*, 2013 [Online]. Available: https://www.cdc.informatik.tu-darmstadt.de/fileadmin/user upload/Group CDC/Documents/Lehre/SS13/Seminar/CPS/cps2014 submission (Vol. 8).

[6] S. Muftic, ‘Bix certificates: Cryptographic tokens for anonymous transactions based on certificates public ledger’, Ledger, vol. 1, pp. 19–37, 2016.

[7] T. Elgamal, ‘A public key cryptosystem and a signature scheme based on discrete logarithms’, *IEEE Trans. on Inf. Th.*, vol. 31, no. 4, pp. 469–472, 1985.

[8] M. O. Rabin, ‘Digital signatures and public-key functions as intractable as factorization’, MIT laboratory for computer science, MIT/LCS/TR-212, Jan. 1979.

[9] R. L. Rivest, A. Shamir, and L. M. Adleman, ‘A Method for Obtaining Digital Signatures and Public-Key Cryptosystems’, *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[10] D. Johnson, A. Menezes, and S. Vanstone, ‘The Elliptic Curve Digital Signature Algorithm (ECDSA)’, *Certicom*, 1998.

[11] B. Preneel, ‘The State of Cryptographic Hash Functions’, *LNCS*, vol. 1561, pp. 158–182, 1999.

[12] P. Paillier and D. Vergnaud, ‘Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log’, LNCS, vol. 3788, pp. 11–20, 2005.

[13] XMPP Standards Foundation, *Extensible Messaging and Presence Protocol*, 2015 [Online]. Available: https://www.xmpp.org/.

[14] BitID, *BitID Open Protocol*, 2015 [Online]. Available: http://bitid.bitcoin.blue/

[15] S. Matsumoto and R. M. Reischuk, *IKP: Turning a PKI Around with Blockchains*, 2016 [Online]. Available: https://eprint.iacr.org/2016/1018.pdf.