Vis-a-Vis: Offline-Capable Management of Virtual Trust Structures Based on Real-Life Interactions

Marco Maier, Chadly Marouane, and Claudia Linnhoff-Popien
Mobile and Distributed Systems Group
Ludwig-Maximilians-Universität München, Germany
{marco.maier, chadly.marouane, linnhoff}@ifi.lmu.de

***Abstract*—Online services, particularly those aimed at a specific user base such as a company's employees, face the problem of identity management. Especially when the service constitutes some kind of social network, i.e., the validity of the users' identities matters, secure and reliable means for identity verification and authentication are required. In this paper, predicated on our previous work, we propose an identity management concept based on a) verification through physical presence and b) authentication through ownership. Our approach being a hybrid solution between a centralized authority and decentralized trust management is settled on a sweet spot between security and convenience for the users. In this extended version, we present the newly proposed Tree of Trust structure in more detail, and provide a thourough explanation how the system can be used in a technically more distributed manner, even supporting offline operation.**

Keywords-identity management systems; authentication; social network services; mobile computing

I. INTRODUCTION

In this paper, we present an extended version of our previously introduced concept called "Vis-a-Vis Verification" [1]. The new additions mainly comprise a more detailed explanation of our trust relationship structure and a completely novel explanation of the offline capabilities and mechanisms of our system.

Nowadays, with about 2.8 billion people using the Internet worldwide [2] and over 1.1 billion people participating in the world's largest online social network Facebook [3], online service providers have a clear need for identity management, i.e., *administration*, *verification*, *authentication* and *authorization* of virtual identities and their real-world counterparts.

Especially when a service's users are linked to their real-world identity (i.e., the service constitutes some kind of online social network) and more so, when the service furthermore requires a high level of security, a key part of identity management is to verify that a virtual account really belongs to the real-world person it is supposed to be linked to, and to provide a secure and intuitive means of authentication. Typically in such services, a user Alice would decide for or against granting certain permissions to a virtual user Bob based on whether she wants to grant those permissions to the real-world Bob. Thus, she has to be sure that the user account really belongs to the real-world Bob (verification), and that nobody else can make requests on behalf of that account (authentication).

There are several ways of verifying a user's real-world identity, which to date either are easy to implement and use but quite easy to attack, or are reasonably secure but introduce a huge overhead in the general process of account creation. In the same way, currently used authentication procedures differ in potential for security breaches on the one, and intuitivity on the other hand.

With the now near ubiquitous usage of smartphones, we see huge potential to improve upon the currently used ways of identity verification and authentication in online services. In this work, we present an approach that is based on two key ideas

- New user accounts are verified to belong to a certain real-world identity by requiring an interaction of an existing user with the new user in the real world.
- The users employ their personal smartphone as the credential for authentication, i.e., the security token is stored on the users' smartphone.

Our approach constitutes a hybrid system. There is a central authority, which is the root of the system's trust relations and is controlled by the organisation employing the system. In order to avoid the typical overhead of sophisticated identity verification, verification tasks are distributed among the system's existing users. Consequently, our system provides a high degree of trustworthiness of the user accounts while keeping the introduced overhead at a reasonable level. To the best of our knowledge, to date, no other approach has settled on that sweet spot between security and ease-of-use.

While the basic design of our system depends on synchronous communication with the central authority, we furthermore developed a more sophisticated approach which enables offline verification of new users.

The rest of the paper is structured as follows. In Section II, we give an overview of various concepts for identity verification and authentication, together with their individual strengths and weaknesses. In Section III, we discuss related work which is or could be used similar to our approach. In Section IV, we present our system for identity verification and authentication. After that, we go into more detail about

management operations within our newly proposed trust relationship structure (Section V). In Section VI, we explain the extended version of our verification procedure which enables offline usage. After that, we describe a real implementation of our concept, which has been deployed for production usage (Section VII). In Section VIII, we describe some scenarios how our approach could be used, and in Section IX, we conclude with an outlook at future work.

II. Identity Management

Identity management of online services comprises several sub-topics like authorization and management of user accounts. The focus of this work specifically lies on *identity verification* and *authentication*. We define identity verification as the process to check the real-world identity of a person and to connect this identity to a virtual account. Authentication then requires some kind of credential to prove that a request is made by that virtual account (i.e., on behalf of the real person).

A. Identity Verification

There are several mechanisms to verify an online identity, i.e., to link a virtual account to a real-world person. These mechanisms can be categorized into three groups, namely *verification through another online identity provider*, *verification through a second communication channel* and *verification through physical presence*.

1) Verification through another online identity provider: The idea of this mechanism is to rely on a third party to verify a new user account. The typical and most widely used example is to require an existing email address when a new account shall be created. To confirm the email address, the online service sends a message to the registrant containing a confirmation link. By clicking the link, the new user can ensure that he is the real owner of the email address. In this case, one relies upon the third party to have checked the identity of the potential user. Thus, it depends on the third party whether the real-world identity is verified, and even if so, typically the real-world identity is not handed over to other parties, leaving the online service with the email address only.

An email address of course is only a very weak personal detail for a real identity. Another approach is to rely on real identity providers. For example, online services like Facebook.com, plus.google.com, or LinkedIn.com manage user profiles, which are verified to some degree. These services can be used either through proprietary interfaces (e.g., *Facebook Login* [4]), or by employing standardised mechanisms like OpenID [5].

Verification through a third party often is the most convenient method of identity verification, both for the end user and the online service provider. The main drawback is the dependence on the trustworthiness of the third party.

2) Verification through a second communication channel: Another approach is the integration of a second communication channel into the verification procedure, typically using an endpoint which requires or inherently is linked to a more sophisticated identity verification like a mobile phone number or a postal address.

When using a mobile phone number, the online service e.g., can send a randomly generated unique token as a text message to the phone. The user then has to enter that token into a form at the online service, which ensures the provider that the user really is the owner of that specific phone number.

A similar procedure can be performed by sending the token in a letter to the user's postal address. Though this alternative takes several days to complete, the online service can obtain a verification of the user's name and residency.

Again, one relies on a third party to verify the identity of a new user. However, e.g., mobile phone providers are required by law to verify the identity of their customers in most countries, leading to a higher trustworthiness of those third parties compared to the previous approach (II-A1).

3) Verification through physical presence: The most sophisticated variant of identity verification is verification through physical presence, i.e., the user whose identity has to be checked is in direct proximity of authorized personnel of the online service provider or a trusted third party which acts on behalf of the provider.

Depending on whether the verifying person already knows the to-be-verified user or not, the new user might have to provide official identity documents like passports or ID cards to prove its identity.

Physical verification by the online service provider itself can be regarded as the most secure option. However, it is often unfeasible to establish a dedicated verification entity at the provider and to manually check the identity of maybe thousands of users. Therefore, services like *Postident* [6] by German logistics company *Deutsche Post* exist, which provide personal identity verification for third parties. In this case, a new user could verify its online account in one of the many stores of the logistics company.

Summing up the alternatives, verification through another online identity provider can be regarded as the most convenient but also most insecure variant. Verification through a second channel like the mobile phone network or old-school snail mail is more reliable due to law-enforced requirements or the sheer characteristics of the channel (e.g., name and postal address is correct when the letter arrives). However, it is also less convenient and more costly for the participants. Finally, verification through physical proximity provides the most secure procedure at the cost of increased effort for both the online service provider and the end user.

B. Authentication

Within the scope of online services, authentication can be defined as the act of confirming the origin of a request, i.e., from which user or account the request was sent. One can distinguish between three categories (*factors*) of authentication, namely authentication by *something you know* (*knowledge*), by *something you are* (*inherence*), and by *something you have* (*ownership*).

1) Something you know: This authentication factor involves some kind of secret only the respective user knows. Typical examples are passwords or pass phrases, personal identification numbers (PIN), or challenge response procedures (i.e., asking a

question only the user can answer). This way of authentication usually can be implemented without much overhead at the provider, but is prone to security breaches resulting from users employing secret credentials too easy to guess or infer from other knowledge. Furthermore, this method can be attacked through phishing [7].

2) Something you are: This means of authentication is based on the behavioral and/or biological characteristics of an individual. Typical methods are to recognize fingerprint, face, voice or retinal pattern. Using inherent characteristics of a human being is convenient for the user because she does not have to remember a secret, but often is complex to implement, error prone and furthermore, the user might be unwilling to share such personal details with a provider.

3) Something you have: In this case, authentication is based on the possession of a key, smart card, security token and the like. In the scope of online services, using this method has the advantage that longer and much more complex security tokens can be used, compared to an ordinary password a user has to know by heart. Implementation usually is straight-forward at the provider, and this method furthermore is very intuitive for the users since it resembles the real-world usage of ordinary keys. However, users might be unwilling to carry additional hardware such as smart cards with them.

Comparing the three methods, authentication based on ownership is the best compromise between security on the one hand, and intuitivity for the users on the other hand. However, using a dedicated hardware component might not be feasible. The latter can be prevented when using a user's smartphone to store the token [8].

C. Problem statement

Today, most online services rely on a verification procedure based on third party identity providers, typically only requiring a valid email address, and employ username-password-credentials for authentication (i.e., something you know). As we have explained, verification through physical presence and authentication via something you have would be a very promising combination regarding security and intuitivity and would therefore be a superior solution to those mechanisms currently most widely used. However, existing ideas result in increased inconvenience for the end-user and more complexity at the provider.

In this work, we present a solution that uses that exact combination of identity verification by physical presence and authentication by something you have, which at the same time keeps the typical overhead at a feasible and usable level.

III. Related work

As seen in the previous section, there is a multitude of ways and combinations online services can perform identity verification and authentication. In this section, we focus on systems that resemble our approach with regard to the employed concepts.

Public Key Infrastructures (PKI) are the most widely used method conceptually comparable to our approach. Digital certificates are issued and verified by a Certificate Authority (CA), which can then be used to authenticate oneself. Dependent on the CA and the type of certificate, obtaining this credential requires the verification of one's real-world identity [9]. PKIs are used in conjunction with Secure Socket Layer (SSL) to ensure secure communication, which in general results in increased complexity leading to vulnerabilities, e.g., with regard to validation of SSL certificates within non-browser environments [10]. However, the main disadvantage is that PKIs in its current form are mostly aimed at organisations and corporations, and distribution of certificates to individual users often is not possible to employ with only a reasonable overhead. Since PKIs allow for hierarchical relationships between the CAs among themselves (i.e., one CA may vouch for another), the resulting structure can be regarded as a tree, which is similar to our approach.

An alternative to the rather centralized trust model of a PKI, which relies exclusively on CAs, is the Web of Trust concept. The latter is a decentralized approach to certificate signing, requiring the users to ensure their respective identities among themselves, often based on personal encounters [11]. PGP and GnuPG are well known implementations of this concept, which allow people to exchange messages securely with mutual authentication [12].

A core concept of the Vis-a-Vis system is the so-called *tree of trust* (see Section IV-D). There are similarly named concepts in other areas which should not be confused with our approach. Presti [13] defines a "tree structure of trust" within the scope of Trusted Computing. In this case, the tree's nodes represent the components of the whole Trusted Computing platform, i.e., from the hardware modules up to the applications. Verbauwhede and Schaumont [14] take a similar approach by partitioning different abstraction levels of electronic embedded systems (e.g., the software level or the circuit level) into secure and non-secure parts. They call the resulting structure a "tree of trust", too. Although both approaches regard trees as a suitable structure for representing trust relationships, they are aimed at different scopes than our system.

IV. Vis-a-Vis

In the following, we describe the Vis-a-Vis concept for identity verification and authentication.

A. Authentication

In order to authenticate the users in the Vis-a-Vis system, a notion of the "something you have" principle is used. The idea is based on the omnipresence of mobile devices such as smartphones or tablets, and the assumption that such devices (or specific accounts on them in case of multi user systems) belong to one and only one user. The device is like a key in the physical world. Authenticating the device therefore suffices to authenticate the respective user.

Technically, authentication is performed by issuing a secret, unique token to each device in the system, which then is included in all requests of the device to the backend (i.e., the provider). To prevent leaking the token, communication between mobile devices and the backend has to be encrypted (e.g., by using SSL). To authenticate the backend itself, traditional means such as SSL certificates can be used.

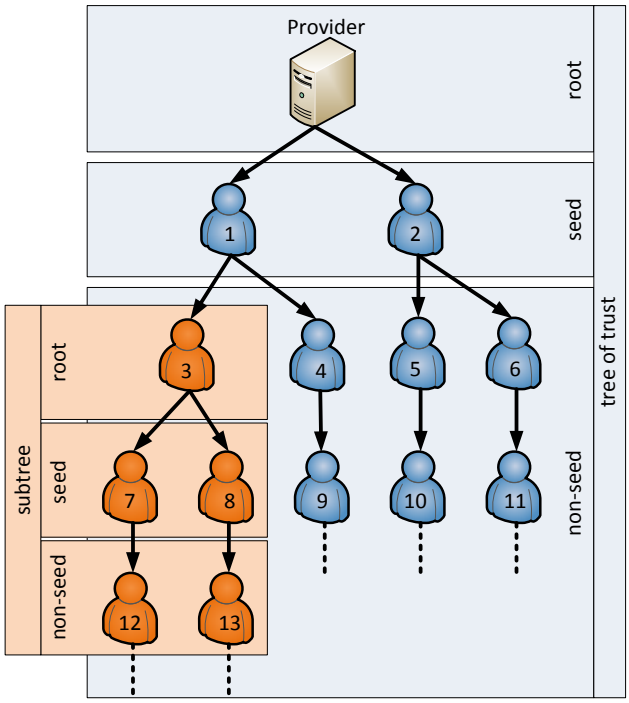

Figure 1. Participants forming a tree of trust, consisting of three levels root, seed and non-seed. Each subtree also is a tree of trust in itself.

B. Participants

Vis-a-Vis is a *hybrid system* with some core components being central elements and most of the other participants self-organizing in a decentralized manner. As such, it is not intended as a single web-wide system but to be deployed individually at organizations. A schematic overview is depicted in Figure 1.

The *Vis-a-Vis provider* is the central entity representing the respective organization. It is fully trusted by default since it manages the whole system. At the moment, there is no interaction beyond provider boundaries and thus, there is no need for further, mutual verification of different Vis-a-Vis providers among themselves.

Providers are responsible to activate *seed users*. These users are verified directly by the provider, by any means regarded secure enough for the given scenario, e.g., by authorized personell such as system administrators verifying a user's identity in person (on-location) or by sending activation information via snail mail. Seed users are fully trusted by the provider.

In order to distribute the verification overhead among the participating entities, seed users can further activate *non-seed users*. The identity of non-seed users is verified by seed users through physical proximity, i.e., seed users may decide to hand over the activation token (from mobile device to mobile device) based on existing knowledge (seed user already knows the new user) or based on official documents (seed user checks, e.g., ID card or passport).

Non-seed users are also allowed to activate new users - in the same manner as seed users - resulting in further non-seed users. As a consequence, non-seed users differ in their distance from the root node (*distance from root*, see Section IV-E), a measure which can be used to quantify the trustworthiness of a user.

C. Protocol

Adding new users to the system is performed in several steps (see Figure 2). First, an online identity (i.e., an account) has to be created for the new user at the provider (step 1). This step can be triggered by the user itself, by the provider (which is reasonable when the future users are known upfront, such as within a company) or by an existing user. It is important to note that in this step, only the account is created (i.e., prepared). It is neither yet activated nor linked to the user's device, i.e., it is not usable, yet.

In order to activate the account, the user needs a one-time key which is generated by the provider. This one-time key can only be given to the new user by the provider itself or by an existing user - the latter case being the more interesting. The new user asks an existing user to verify her identity (steps 2 and 3). The existing user wanting to activate the new user requests the new user's one-time key from the provider (steps 4 and 5) and then forwards it to the new user (step 6). The forwarding has to be done in a way requiring physical proximity (i.e., "vis-a-vis"), e.g., transfer via Near Field Communication (NFC) or optical codes like QR codes.

After receiving the one-time key, the new user sends the key directly to the provider (step 7). The provider now checks whether it is the correct key for the respective user and, when confirmed, sends an authentication token back to the new user (step 8). The user includes this token in all subsequent requests to the backend to confirm their authenticity (step 9).

D. Tree of trust

Performing the above protocol using the described participants results in a tree-like structure. Since this structure describes the evolved trust relations between the users, we can formally define a *tree of trust*

$$T = (V, E) \tag{1}$$

with nodes V and edges E as a rooted tree with *root node* $r \in V$ (the Vis-a-Vis provider), an arbitrary number of *seed nodes* (seed users)

$$S = \{s : s \in V \wedge (r, s) \in E\} \tag{2}$$

and an arbitrary number of *non-seed nodes* (non-seed users) $\bar{S} = V \setminus S$. Each *rooted subtree*

$$T' = (V', E') \tag{3}$$

with $E' \subseteq E$ and $V' = \{v' : v' \in V \wedge (\exists v'' \in V' : (v'', v') \in E' \vee (v', v'') \in E')\}$ is also a tree of trust, i.e., each node can be regarded as the root of its own tree of trust containing users which have been activated by itself or its descendants.

Trees of trust are an analogy to the idea of the web-of-trust. The difference is that trees of trust represent a hierarchy of users allowing for a more intuitive assignment of capabilities with regard to some metric (see Section IV-E) whereas in a meshed graph the structure of trust relationships is harder to grasp.

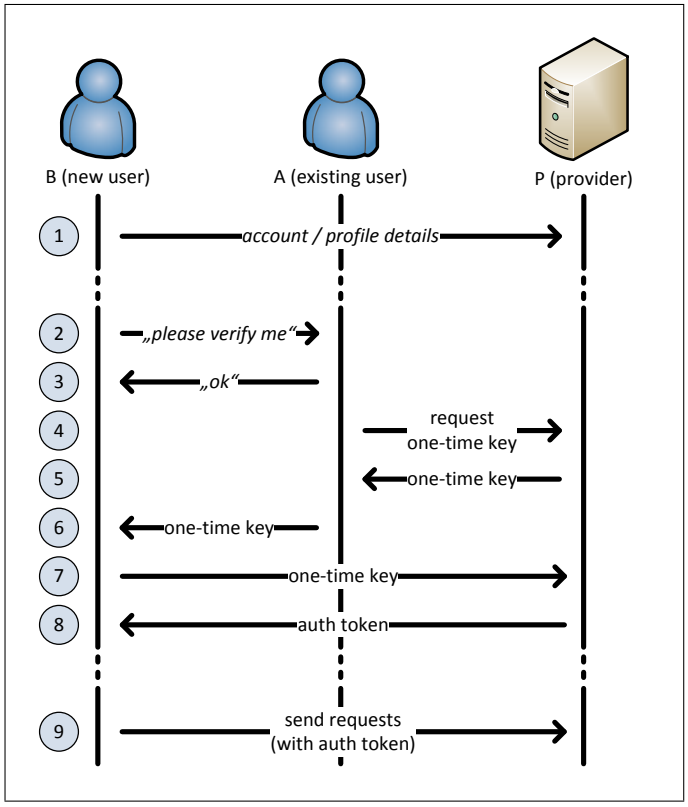

Figure 2. The Vis-a-Vis protocol.

E. Distance from root D_r

There is a single path $P(x, y)$ between each two nodes x and y in the tree, defined as

$$P(x, y) = (v_1, \ldots, v_n) \tag{4}$$

with $v_i \in V, v_1 = x, v_n = y, (v_i, v_{i+1}) \in E$. Based on that we define a measure *distance from root* D_r as

$$D_r(v) = |P(r, v)| \tag{5}$$

A user's distance from its tree's root is a measure for the user's trustworthiness. This measure can be considered when assigning rights or capabilities, e.g., one might limit the length of an *activation chain*, i.e., the path from the tree's root to the user, to a constant C, i.e., $\forall v \in V : D_r(v) < C$.

F. Weighted Tree of Trust

Often it might be desirable to establish a more flexible scheme to assign a trust value to the nodes, considering not only the length of the path from them to the root node but also impact factors like the trustworthiness of the used activation channel.

Furthermore, in some scenarios it is useful to not regard the users as the tree's nodes but their individual devices. Users often possess several mobile devices and it is advisable to issue individual authentication tokens to each device. In case a token is compromised, one can revoke the token without affecting the user's other devices.

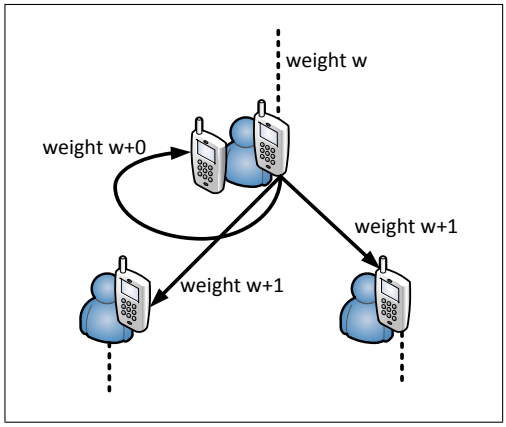

Figure 3. Part of a weighted tree of trust, showing activation of new users (with decreasing trustworthiness) as well as self-activation with edge weight 0 (i.e., no loss of trustworthiness).

Activating a new device by oneself would reduce the trust value of the new device when using the distance from root measure. This can be the desired behaviour, but more often the same person should have the same capabilities on each of its devices.

This problem is solved by introducing weights on the tree's edges (see Figure 3), i.e., each edge (x, y) is assigned a weight w_{xy} correlating to the trustworthiness of the edge itself. Thus, one can define a new trust measure $Trust$ as

$$Trust(v) = \sum \{w_{v_i v_{i+1}} : (v_i, v_{i+1}) \in P(r, v)\} \tag{6}$$

When setting the weight of all edges to 1, $Trust(v) = D_r(v)$.

Using the $Trust$ measure one can allow activation of one's own devices without loss of (calculated) trustworthiness by setting the edge weight to 0. On the other hand, one can also assign edge weights > 1 to mark "more insecure" activations.

V. Tree of Trust Operations

In order to build and maintain the Tree of Trust structure, several operations are required for handling certain events such as a new user joining the system. These basic operations are *adding*, *removing*, *promoting* and *demoting* nodes. For most of these operations there is no definitive way how they should be done, they rather depend on the given scenario. In the following, possible behaviors are described, which span most of the intended use cases.

A. Adding

When a new user gets activated, i.e., an existing user or the provider has verified her identity by performing the Vis-a-Vis protocol, a node corresponding to the user (or rather the user's device) is added to the tree of trust. In case of an unweighted tree, simply adding the node is sufficient. In case of a weighted tree, one has to determine the weight to be assigned to the newly introduced edge in the tree. The weight, e.g., might be dependent on the trustworthiness of the employed channel for performing the verification procedure or the participating users (e.g., when a user activates another device of herself, the weight typically is 0).

B. Removing

The tree of trust itself is not meant to be a structure to manage a system's users themselves but rather their trustworthiness. Thus, it is not required per se to remove inactive users from the tree because the trustworthiness of a user often is not affected by the other users' status. However, oftentimes it might make sense to keep the tree of trust nodes in sync with the system's current state of active and inactive/removed users. Furthermore, in case a device (or its access token) gets stolen, it is required to remove the corresponding node from the tree of trust.

When a node is removed from the tree, it has to be taken care of its child nodes. There are two cases: i) the node has been removed because it has been compromised, i.e., its child nodes' trustworthiness might be affected, and ii) the removed node was in a secure state, i.e., its child nodes' trustworthiness is not affected.

In case i), the child nodes have to be removed recursiveley as well because they might have been activated by an unauthorized user. In order to prevent removal (i.e., deactivation) of a lot of nodes which might have been activated before the node was compromised, it is recommended to store a timestamp of the node's activation, so that only child nodes get removed which have been activated after a given date (e.g., the last time the node is known to have been in an uncompromised state). The other nodes then can be treated like in case ii).

In case ii), child nodes should be kept in the tree and should retain their assigned trustworthiness. This can be accomplished in two ways. One approach is to keep a placeholder of removed nodes in the tree. Such nodes are called *ghost nodes*. They cannot perform any further actions, but are left in the tree to preserve the value of trust calculations for its child nodes. The other way is to reassign child nodes to another parent node. This theoretically can be any node, but usually should be the parent node n_P of the removed node n_R itself. In order to preserve the trust value of a given child node n_C, the newly introduced edge (n_P, n_C) has to be assigned the sum of the weights of the edges (n_P, n_R) and (n_R, n_C), i.e.,

$$w_{n_P,n_C} = w_{n_P,n_R} + w_{n_R,n_C} \tag{7}$$

The first approach has the advantage to retain the activation chains that really happened and it can be applied to unweighted trees as well. The second approach on the other hand does not need to keep track of ghost nodes, but is only applicable to weighted trees.

C. Promoting

It can happen that a user (i.e., her device) gets activated a second time, maybe by a user at a higher level in the tree of trust (leading to a promotion of the user, i.e., an increased trustworthiness). This facilitates self-organization of the userbase, since users might first be verified by the "next best" user (fast activation) and then at a later point in time be reactivated by another user to gain a higher trust value. Analogly, also the weight of an edge could be decreased, which can be reduced to a user being reactivated by the same parent in a more trustworthy manner. The question again is how child nodes should be treated in such events.

The simplest way is to transitively accept increased trust values for all of the node's children as well, i.e., the reactivated node gets reassigned to the new parent node leading to the whole subtree being moved within the tree. Most of the time, this is a reasonable approach.

However, it might also be the case that the child nodes' trustworthiness should not be affected by a promotion of their parent. In this case, the reassignment of a node can be reduced to removing and then adding it again (see Sections V-A and V-B). In case one opts for the ghost node approach, it might be reasonable to keep a reference from the ghost node to the actual node's new location in the tree. When using the second approach, the new parent of the child nodes of course should be the existing parent at its new location in the tree, however, with the weight of the connecting edge adjusted so that the child nodes preserve their previous trust value.

D. Demoting

As a counterpart to promoting, nodes might also get demoted, i.e., reassigned to a parent farther down the tree of trust or the edge weight might be increased. In general, one can treat this event in the same ways as a promotion of a node.

However, demoting a node might affect existing activation chains. i.e., it could be that some activations would have been prohibited by given rules or constraints (e.g., farthest distance from root for new activations) if the demotion had happended earlier. In case the demotion is intentional because of, e.g., the previous trust value being higher than what is reasonable for a given node, it might be required to deactivate and remove certain child nodes.

Depeding on the specific rules and settings in a given system, whenever the trust value of a non-leaf node changes (i.e., in case of promotion or demotion), the corresponding subtree might have to be examined recursively to maybe alter values or status of edges or nodes.

VI. Offline usage

So far, the Vis-a-Vis protocol for account verification and activation (see Section IV-C) depends on a synchronous procedure, i.e., both the existing user and the user to be activated have to be connected to the system, i.e., the Vis-a-Vis provider. We found this to be a rather fierce requirement, especially in one of our intended deployment scenarios (see Section VIII-B).

In this section, we will introduce an extension to the Vis-a-Vis protocol, which allows for offline verification of new users, without synchronously communicating with the Vis-a-Vis provider. The extended protocol even enables transitive activations without requiring communication with the provider until a given account wants to use the service for the first time.

In its basic version, the Vis-a-Vis protocol requires that the account (i.e., profile information) for the new user is created at the provider upfront. This can happen in one go with but can

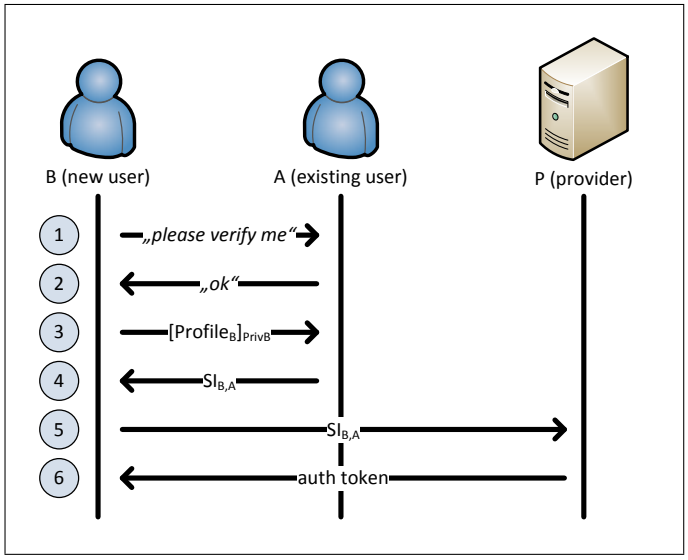

Figure 4. The extended Vis-a-Vis protocol, allowing offline usage.

also be decoupled from the verification itself. In order to activate a given account, the new user then has to be handed over a one-time key from the provider via an existing (authorized) user. This second step requires synchronous communication with the provider and thus has to be replaced by an offline capable mechanism. This mechanism is based on public-key cryptography and is explained in the following.

A. Preliminaries

In the extended activation process, the participating entities again are the provider P, an existing user A and a new user B. P owns a key pair consisting of the private key Priv_P and the corresponding public key Pub_P. The latter has to be known to all participating entities.

Just as before, P stores user profiles containing *Personally Identifying Information (PII)*, such as forename, surname and date of birth. In this extended variant, each user profile furthermore contains the user's public key, i.e., the profile of user A is defined as the tuple:

$$\mathrm{Profile}_A = (\mathrm{Forename}_A, \mathrm{Surname}_A, \mathrm{DateOfBirth}_A, \mathrm{Pub}_A) \quad (8)$$

Of course, A itself has to be in possession of the corresponding private key Priv_A. Essentially, every user in the system owns her own key pair, which then is used to authenticate messages with the help of digital signatures. In the following, the notation $[\mathrm{Data}]_{\mathrm{Priv}_X}$ means that the data packet Data is signed with the private key Priv_X, i.e., its authenticity can be verified with the corresponding public key Pub_X. $[\mathrm{Data}]_{\mathrm{Priv}_X}$ consists of Data itself plus the computed signature $\mathrm{Sig}_{\mathrm{Priv}_X}(\mathrm{Data})$.

B. Offline Verification

The basic idea of the extended protocol is to digitally sign the profile information $\mathrm{Profile}_B$ of a new user B with a private key of which the provider already knows the corresponding public key. The provider then can check that the profile information were verified by an existing user in his system. The extended protocol is depicted in Figure 4.

The process starts with B generating his own key pair consisting of Pub_B and Priv_B and then creating the data packet $[\mathrm{Profile}_B]_{\mathrm{Priv}_B}$. When meeting a suitable existing user A (steps 1 and 2), B transfers this data packet to user A (step 3). The transfer can happen through any communication channel but typically one would employ the same means of data exchange as used for real vis-a-vis transmissions, such as optical codes or NFC.

User A first has to check the validity of the signature of the received packet to make sure that B really is in possession of the corresponding private key of the public key Pub_B, which is included in $\mathrm{Profile}_B$. In case the signature is valid, A signs $\mathrm{Profile}_B$ with his own private key, i.e., A creates the *signed identity*

$$\mathrm{SI}_{B,A} = [\mathrm{Profile}_B]_{\mathrm{Priv}_A} \quad (9)$$

$\mathrm{SI}_{B,A}$ then has to be handed over to user B (step 4). This step now is required to be performed vis-a-vis just like the transmission of the one-time key in the basic protocol (see Section IV-C). It is A's responsibility to transfer the signed identity to the real B only and no one else. So far, the whole process can be performed offline without any interaction with the Vis-a-Vis provider.

Whenever B now is able to establish a connection to the Vis-a-Vis provider, she can authenticate and register herself with the help of the signed identity. To do so, she sends $\mathrm{SI}_{B,A}$ to the provider (step 5). The latter can validate the signature with the help of the users' public keys stored in its database. In case the signature is valid, the provider adds $\mathrm{Profile}_B$ to its database and provides B with a new access token she can use to authenticate herself when accessing any services of the system (step 6).

C. Pending Chain

Regarding the above process, a question may arise: What if B verifies another new user C before B itself has sent its signed identity to the provider, i.e., the provider does not know that B is a valid user? In this case, new accounts such as C arrive in a *pending state*. As C might also verify further users, it is possible for longer *pending chains* to come into existence. Being *pending* means that the provider has received the signed identity of a new user and has provided the user with a (maybe temporary) access token, but has not yet granted any permissions to the new user because there are users in the pending chain, which have not yet presented a valid signed identity to the provider. Whenever the provider receives a new valid signed identity, it has to check all pending users if their pending state can be resolved now.

D. Distributed Permissions

The pending chain essentially is another component in the verification and activation procedure which somehow blocks the process, preventing new users from accessing the system

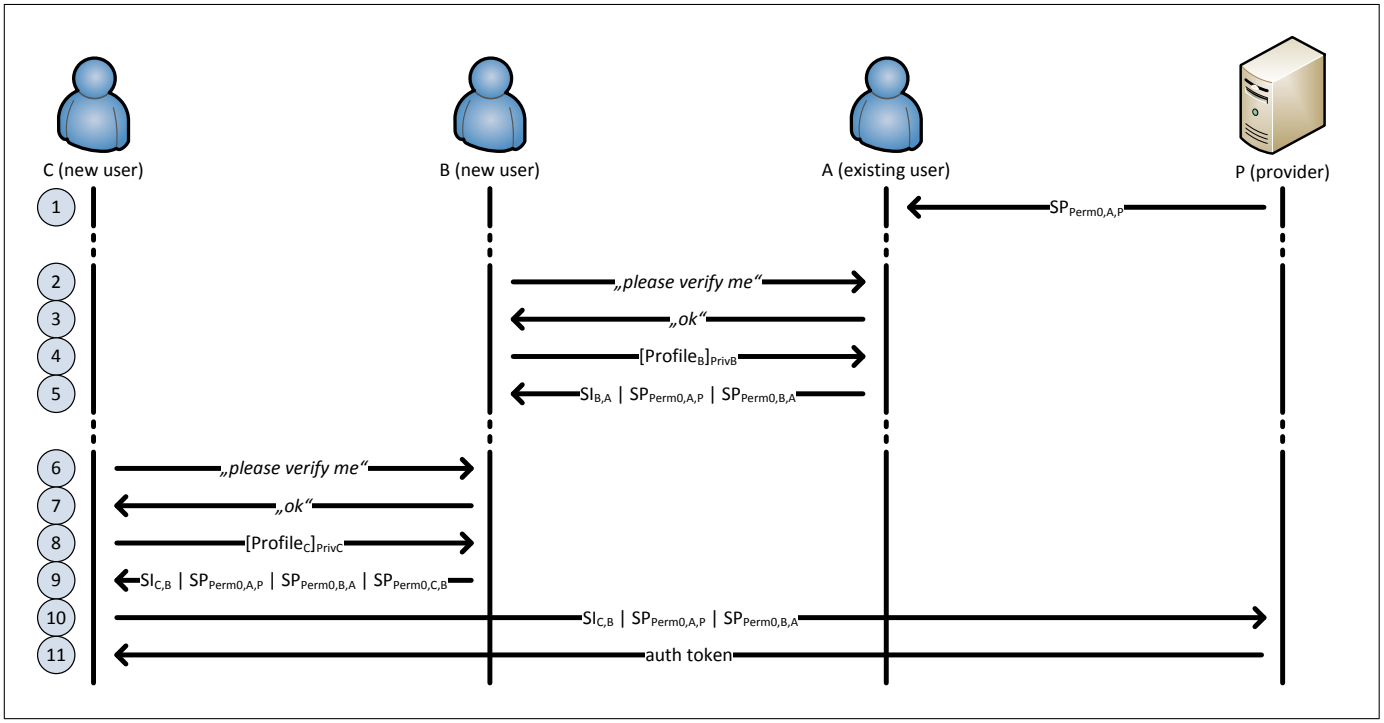

Figure 5. The extended Vis-a-Vis protocol, which is offline-capable and includes distributed permissions management.

before every previous user in the pending chain has been activated. Furthermore, new users cannot make sure that the user who verifies them really is a legit user of the system. In order to solve both these problems, we introduce the concept of *distributed permissions*.

We define a set of permissions

$$\mathrm{Perms} = \{\mathrm{Perm}_0, ..., \mathrm{Perm}_n\} \tag{10}$$

and for each permission Perm_i, we define a set of *prerequisites* $\mathrm{Prereqs}_{\mathrm{Perm}_i} \subseteq \mathrm{Perms}$.

In the most basic case, there is only one permission Perm_0 with the meaning "the user may verify further users". The prerequisite for Perm_0 to be valid is that the issuing entity also has the permission to verify further users, i.e., $\mathrm{Prereqs}_{\mathrm{Perm}_0} = \{\mathrm{Perm}_0\}$. We define a *signed permission* as

$$\mathrm{SP}_{\mathrm{Perm}_i,X,Y} = [(\mathrm{Perm}_i, \mathrm{Pub}_X)]_{\mathrm{Priv}_Y} \tag{11}$$

meaning user Y is issuing permission Perm_0 to user X (represented by her public key Pub_X).

The complete verification procedure with included distributed permissions management is depicted in Figure 5. Let A again be an existing user in the system who should be able to verify new users. A therefore is issued a *signed permission* $\mathrm{SP}_{\mathrm{Perm}_0,A,P}$ by the provider (step 1). When A wants to verify a new user B, she basically follows the procedure as described in Section VI-B but not only hands over the signed identity $\mathrm{SI}_{B,A}$ but also the signed permission $\mathrm{SP}_{\mathrm{Perm}_0,A,P}$ to B. In case B herself should be allowed to verify futher users, A also generates another signed permission $\mathrm{SP}_{\mathrm{Perm}_0,B,A}$ and provides it to B (steps 2 to 5).

As explained in the beginning, every user in the system knows the providers public key Pub_P. B therefore now can validate whether A is a legit user who is allowed to verify further users by checking the signature of the signed permission $\mathrm{SP}_{\mathrm{Perm}_0,A,P}$ with the help of Pub_P and by checking the signature of $\mathrm{SI}_{B,A}$ with the help of Pub_A, which is included in the signed permission. The latter is necessary to make sure that A really is in possession of the private key Priv_A, which corresponds to the public key Pub_A to which the permission Perm_0 was issued.

B now can verify another user C by following the same procedure as above (steps 6 to 9). However, B has to provide several information packets to C, which are needed to represent the current verification state:

1) $\mathrm{SI}_{C,B}$
2) $\mathrm{SP}_{\mathrm{Perm}_0,A,P}$
3) $\mathrm{SP}_{\mathrm{Perm}_0,B,A}$
4) $\mathrm{SP}_{\mathrm{Perm}_0,C,B}$ (optional)

Based on these information, both C itself as well as the Vis-a-Vis provider are able to validate the correctness of C's verification, even without B being activated yet. To do so, one has to recursively check the provided information. First, the signature of the signed identity $\mathrm{SI}_{C,B}$ is checked with the help of public key Pub_B, which is included in the signed permission $\mathrm{SP}_{\mathrm{Perm}_0,B,A}$. Then, it has to be made sure that Pub_B belongs to an entity which is allowed to verify new users, i.e., which has been issued the permission Perm_0. Therefore, the signature of the signed permission $\mathrm{SP}_{\mathrm{Perm}_0,B,A}$ has to be checked with the help of the public key Pub_A,

which is included in the signed permission $\mathrm{SP}_{\mathrm{Perm}_0,A,P}$. Since A is another user and not the provider itself, it now has to be checked whether A has the preconditional permissions to issue Perm_0. In this case, this means that A must also have been issued permission Perm_0. The latter can be validated by checking the signature of the signed permission $\mathrm{SP}_{\mathrm{Perm}_0,A,P}$ with the help of the pre-shared public key Pub_P of the provider.

By including the complete history of issued permissions in a given verification chain (which can be of arbitrary length), the validity of a user verification can be checked independently of the state of previously verified users. Thus, upon receiving and validating such a verification chain (step 10), the provider may hand out an access token to any verified user, no matter if all the other users in the verification chain have been activated yet (step 11). Futhermore, each user who gets verified by another user can also check by herself whether the verification was valid or not, thus preventing misuse of the offline verification feature.

The usage of signed permissions also enables more complex permission sets. One use case would be to limit the length of verification chains, e.g., that only chains of maximum length 2 are allowed. In this case, one could define the following permissions:

- Perm_0: "the user may verify further users"
- Perm_1: "the user may issue Perm_0"
- Perm_2: "the user may issue Perm_1"
- $\mathrm{Prereqs}_{\mathrm{Perm}_0} = \{\mathrm{Perm}_1\}$
- $\mathrm{Prereqs}_{\mathrm{Perm}_1} = \{\mathrm{Perm}_2\}$

An existing user A then might be issued Perm_0 and Perm_1 by the provider. A now can verify another user B because he has permission Perm_0. He furthermore can issue Perm_0 to the new user B because he has permission Perm_1. Consequently, B can verify another user C. However, B cannot issue Perm_0 to another user because B does not have permission Perm_1. Perm_1 can only be issued by someone having permission Perm_2, which in this case is limited to the provider itself, thus preventing longer verification chains.

VII. Implementation

We have implemented and deployed the proposed concept in a real production environment at an educational institution. In this section, we will briefly describe the technical implementation of the various components.

The technical part of the Vis-a-Vis provider has been realised as a backend service, which is programmatically accessed through a REST interface. It furthermore provides a web interface, which is intended for account creation. We employ a weighted tree of trust (see Section IV-F), i.e., regarding the users' devices as the tree's nodes and allowing for self-activation of more than one device. Devices are running a custom application, which stores the authentication token and furthermore is used to access protected content provided by the institution.

When a new user wants to create an account, she does so using a dedicated account creation web interface of the Vis-a-Vis provider. Thereby, the user has to provide some personal credentials like name and date of birth, as well as her affiliation to certain groups or departments of the institution. When submitting the registration request, a QR code containing a unique account ID is shown. The user has to scan the code with her smartphone running our custom application, which results in an association of the user's device to the newly created account. It has to be noted that at that point in time, only the association is created, the account itself is not activated, i.e., the user cannot access any protected content, yet.

After that, the user has two choices. She either proceeds to print out a document containing her account credentials including the associated account ID. She then has to sign the document and to provide it to authorized personnel at the institution. The latter now check the provided credentials, verify the identity of the new user and then can activate the associated account. The user now can access the protected content and has become a seed user, as she was verified by the Vis-a-Vis provider itself. The seemingly cumbersome usage of printed documents is introduced because at the given institution, it is legally required that the to-be-created seed users sign a consent form. Thus, the Vis-a-Vis system is integrated into the existing workflow.

The alternative way of activating an account is via an existing user. The system is configured to allow existing users to activate new users which belong to the same group. In our mobile application, existing users can browse through and select users which they can activate. They can request the needed one-time key from the provider, which then is encoded in a QR code. The new user can scan this code, resulting in the described protocol being carried out (see Section IV-C). Consequently, the new user has become a non-seed user.

VIII. Applications

The Vis-a-Vis concept is predestined to be used at any organisation with a hierarchical structure such as companies, educational institutions, clubs or small project teams. In the following, we describe two use cases, in which our system perfectly fits the inherent structure of the scenario.

A. Use in Companies

A company usually is organised in a hierarchical way, composed of departments and teams, where permissions often should be assigned in accordance to that structure. This perfectly fits the basic building blocks of the Vis-a-Vis system, where senior employees might activate other employees. The hybrid approach of the Vis-a-Vis system ensures that some kind of central authority is present and thus, that seed users can be trusted. Each principal of the respective hierarchy level acts as the responsible seed user of his subordinates. As an example, the CEO of a company would act as a main seed user and unlock its subordinate head of department. In the following, department heads can activate their subordinate team leaders, and so on.

The resulting tree of trust can be used to assign permissions and capabilities, not only based on the user's role but also on her distance to the last directly verified user (which can be measured by the distance from root metric).

B. Use in schools

Another interesting use case is constituted by educational institutions, e.g., schools. This in fact is the scenario in which we have already deployed the system. Within a school, several roles exist, such as teachers, students and parents. These roles are subject to a predetermined hierarchy with different permissions. Furthermore, it is of highest relevance that user identities are verified, i.e., parents and teachers can be sure that they are corresponding with each other.

In this case, initially only the director of a school might have access to the system. As a director representing the highest authority within the school, he has the ability to unlock teachers as seed users. These in turn have the privilege to unlock students who belong to their assigned classes. Students can then activate their parents and give them the permission to access the school network, too.

A key benefit in this use case is the decreasing administrative costs because of the convenient but secure delegation of activation responsibilities.

In case a written agreement from the parents is required by law, the Vis-a-Vis concept is also employable, with parents being authorized directly by the school management (and therefore becoming seed users). Parents then are able to activate further family members by themselves.

In some of our real deployment environments in Germany we found that the school building (sometimes intentionally) is constructed to attenuate or even completely shield mobile network signals, in order to prevent mobile phone usage during lessons and tests. Wifi networks typically are reserved for faculty members. Consequently, our basic (synchronous) protocol was not usable in these environments. We therefore developed the extended protocol as described in Section VI, which allows to verify student accounts within the school building even while being offline. The students later can complete the registration procedure after school, when they are back to having internet access.

IX. Conclusion and future work

In this paper, we presented a novel approach to combine the concept of identity verification through physical presence with the authentication factor ownership, i.e., authentication by something you have. We defined a structure called tree of trust, on which a distance from root metric can be calculated. The latter is a measure for a node's trustworthiness, i.e., it can be used as a parameter for permission assignment. By extending the concept to weighted trees of trust, one can also allow for self-activation of further devices as well as activation by more insecure means, resulting in a lower trustworthiness value. We described several ways for adding, removing, promoting and demoting nodes in the tree, which shows its applicability to various use cases. The system perfectly fits scenarios which inherently exhibit some kind of hierarchy and require a central authority, but in which identity verification tasks should be distributed among the system's users.

In order to allow distributed verification even without internet connection (i.e., without being able to communicate with the central authority), we presented an extended protocol including distributed permissions management, which allows for offline usage of main parts of the Vis-a-Vis system. The distributed permissions system is flexible enough to even allow complex permissions such as to limit the verification chain to a certain maximum length.

In future work, it will be interesting to investigate the integration of proximity proofs, i.e., to check whether the transmission of the one-time key really has taken place vis-a-vis, i.e., in direct physical proximity. This would further increase the system's security and the reliability on the trustworthiness of activated accounts. It is of even higher interest in the case of our extended protocol. The necessary information to represent longer verification chains (including public keys, etc.) can become too large to be encoded in optical codes and NFC is still not supported by lots of devices. Thus, it might be required to offload data transfer to communication channels such as Bluetooth, which has a too long range to be called a vis-a-vis channel. Integrating proximity proofs would greatly improve the system's trustworthiness in this case.

We are furthermore investigating how the offline-capable and distributed trust and permissions features could be extended to allow for peer-to-peer operation in order to completely omit a central authority.

References

[1] M. Maier, C. Marouane, C. Linnhoff-Popien, B. Rott, and S. A. Verclas, "Vis-a-vis verification: Social network identity management through real world interactions," in SOTICS 2013, The Third International Conference on Social Eco-Informatics, 2013, pp. 39–44.

[2] International Telecommunications Union, "Key ict data for the world," http://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx, 2013, last accessed date: 2014-05-31.

[3] "Facebook key facts," http://newsroom.fb.com/Key-Facts, 2013, last accessed date: 2014-05-31.

[4] "Facebook login," https://developers.facebook.com/docs/facebook-login/, 2013, last accessed date: 2014-05-31.

[5] "Openid," http://openid.net/, last accessed date: 2014-05-31.

[6] "Postident," http://www.deutschepost.de/de/p/postident.html, last accessed date: 2014-05-31.

[7] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell, "Client-side defense against web-based identity theft," in Proceedings of the 11th Annual Network and Distributed System Security Symposium, 2004.

[8] T. V. N. Rao and K. Vedavathi, "Authentication using mobile phone as a security token," International Journal of Computer Science & Engineering Technology, vol. 1, no. 9, pp. 569–574, October 2011.

[9] G. Coulouris, J. Dollimore, and T. Kindberg, Distributed Systems: Concepts and Design (4th Edition) (International Computer Science). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[10] M. Georgiev et al., "The most dangerous code in the world: validating ssl certificates in non-browser software," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 38–49.

[11] J. Golbeck, B. Parsia, and J. Hendler, "Trust networks on the semantic web," in Cooperative Information Agents VII, ser. Lecture Notes in Computer Science, M. Klusch, A. Omicini, S. Ossowski, and H. Laamanen, Eds. Springer Berlin Heidelberg, 2003, vol. 2782, pp. 238–249.

[12] P. R. Zimmermann, The official PGP user's guide. Cambridge, MA, USA: MIT Press, 1995.

[13] S. L. Presti, "A tree of trust rooted in extended trusted computing," in Proceedings of the Second Conference on Advances in Computer Security and Forensics Programme, 2007, pp. 13–20.

[14] I. Verbauwhede and P. Schaumont, "Design methods for security and trust," in Design, Automation & Test in Europe Conference & Exhibition, 2007, pp. 1–6.