# Security Analysis of Leap-of-Faith Protocols

Viet Pham[1] and Tuomas Aura[2]

[1] Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK
viet.pham.2010@rhul.ac.uk
[2] Aalto University, P.O.Box 15400, FI-00076 Aalto, Finland
tuomas.aura@aalto.fi

**Abstract.** Over the Internet, cryptographically strong authentication is normally achieved with support of PKIs or pre-configured databases of bindings from identifiers to credentials (e.g., DNS to public keys). These are, however, expensive and not scalable solutions. Alternatively, Leap-of-Faith (LoF) provides authentication without additional infrastructure. It allows one endpoint to learn its peer's identifier-to-credential binding during first time communication, then stores that binding for future authentication. One successful application of LoF is SSH server authentication, encouraging its introduction to other protocols.

In this paper we analyze the security of LoF protocols. Various aspects are discussed to show that several proposed LoF protocols have weaker security than SSH, and that their security also depends on design and implementation details. Several protocols were analyzed, including SSH, TLS, BTNS, and HIP, revealing attacks such as impersonation, man-in-the-middle attacks, and credentials flooding. Consequently, additional mechanisms and best practices are proposed to strengthen LoF applications.

**Keywords:** leap-of-faith, authentication, key management, SSH, TLS, BTNS IPsec, HIP, decentralized system, infrastructureless.

## 1 Introduction

Due to physical separation, Internet communication suffers from identity-spoofing attacks, such as impersonation and man-in-the-middle (MitM). When two parties communicate, they need a way to name each other. Each party is represented by a *communication identifier*. For example, in `telnet` remote login, the communicating parties are the server and the client user (not the client computer). The server is identified by its DNS name or IP address, whereas the client user is identified by a username. However, since `telnet` transmits username and password in plaintext across the Internet, these identifiers could be easily spoofed.

To prevent identity-spoofing attacks, there are authentication methods based on *cryptographic credentials*. Each credential is owned by one entity and can be used to verify its identity. For instance, in public-key authentication, public keys are used as credentials. To facilitate authentication, the identifier of an entity must be mapped to that entity's credential. When someone claims to have an

M. Rajarajan et al. (Eds.): SecureComm 2011, LNICST 96, pp. 337–355, 2012.

identifier, the authenticator can use the corresponding credential to verify the ownership. The main problem with authentication is to maintain such kind of identifier-to-credetial mappings, or *security bindings* in our terminology.

Authentication should be strong in the sense that all bindings accepted by the authenticator are correct. In distributed systems, strong authentication is usually supported by a trusted third party (TTP) or a public-key infrastructure (PKI). For example, in TLS, each binding is represented by a certificate. Each certificate must be signed by a certificate authority (CA) within the PKI hierarchy, and its correctness could be securely verified given the public key of the root CA. Similarly, symmetric-key systems like Kerberos provides strong authentication [1] using a key distribution center (KDC) as a TTP. The strong authentication with a PKI or TTP, however, does not come without costs:

- Registration effort: correct bindings must be registered with the CA or TTP, which requires administrative effort [2]. For example, the TTP administrator must carefully perform background check on the binding owner, or otherwise attacks are possible, e.g., [3].
- Cost: as a business process, each registration incurs a cost to the registering party to have its binding certified. Most individuals and many businesses are unwilling to pay such fees especially for local or temporary IT systems.
- Scalability: with the current size of the Internet, no TTP is capable of maintaining a database of bindings for every network entity. This is especially true in peer-to-peer communication in which all endpoints are equal the their number can be very large.

These limitations have led to a search for alternative forms of authentication. One possibility is *recommendation systems* with PGP Web of Trust [4] as an early example. In these systems, the reliance on the trustworthiness of CAs is replaced by trust between people based on experiences and recommendations. As certification of bindings is decentralized to a community rather than a single PKI hierarchy, the registration and management costs could be lower. However, modelling of trust in these systems requires complicated mathematical techniques (e.g. [5]) and their applicability to non-human entities like computers is still unclear. Also, both the authenticator and the peer being authenticated must be in the same community, making it less scalable globally.

Another idea is the use of *self-certifying identifiers*. An identifier is generated by evaluating some collision-resistant function $f$ on a credential, so that the mappings between identifiers and credentials are one-to-one, hence avoiding impersonation. As identifiers can be generated locally, no third party is needed and there is no administrative overhead. This idea works well when identifiers can include arbitrary bit strings, for instance, in the SEND protocol [6] with *cryptographically generated address* [7,8]. In contrast, user-interactive applications require human-readable identifiers, which cannot be arbitrary hash values. *Identity-based cryptography* [9] attempts to solve this, but still requires a TTP, i.e., a private key generator and thus suffers from similar problems as PKI.

In this paper, we focus on the *Leap-of-Faith (LoF)* method as an alternative to strong authentication. LoF is familiar to most university users from SSH server

authentication, which can be used without pre-distributed keys or certificates. This paper is motivated by the fact that there are many proposals to imitate the SSH authentication in many other protocols. Our goal is to understand why the relatively weak LoF principle has been successful in SSH and whether it will readily extend to other protocols that operate with slightly different assumptions. This kind of analysis is important because security mechanisms often fail when they are taken outside their original operating environment.

In section 2 we describe the LoF principle in detail. Then, section 3 discusses several security considerations associated with LoF. These considerations are applied in section 4 to 7 to analyze the security of several protocols that make use of LoF. Evidently, we present a number of weaknesses on these protocols. Finally, we propose in section 8 a number of mechanisms to strengthen the security of this authentication method.

## 2 Leap of Faith

Due to aforementioned limitations, strong authentication is sometimes not feasible. Arkko and Nikander [10] report the emergence of new *weak or infrastructureless authentication* techniques as a workaround. The weak techniques could be used in settings where there is no sufficient information or infrastructure to establish a trust chain for strong authentication [10, 1]. One such technique is the LoF mechanism, which can be summarized as follows:

- *First communication and leap of faith*: When the authenticator and its peer communicate for the first time, the authenticator is unable to securely authenticate the peer. It takes the identifier and credential presented by the peer, checks that the identifier is new, and locally stores this identifier-to-credential binding.
- *Subsequent communication*: In any subsequent communication between the authenticator and the same peer, the communication channel is authenticated using the stored binding.

The security of LoF relies on an important assumption that the attacker is unlikely to be present during the first communication. If the accepted binding is correct, all subsequent communication will be secure. Clearly, LoF does not provide strong security and the level of assurance depends on the details of the communication network and the types of attackers that one is trying to defend against.

In the classical computer security model, LoF is unquestionably insecure. In practice, it has been applied in several contexts. The most prominent one is SSH. Another common situation is access to web sites whose TLS certificate is not signed by a recognized CA and the client user chooses to store it for future authentication. Also, when a user downloads and installs a web browser or an operating system, a list of root certificates or code signing keys is configured on that user's machine. Most users do not bother to verify offline the correctness of these certificates, and thus the LoF mechanism is applied.

It is also worth comparing LoF to a related idea, *first-come-first-serve allocation* of identifiers. For example, free email service and online forums allow new users to create a name-to-credential binding for any name that has not been previously allocated. That part is similar to LoF. The difference is that, in LoF, there are some external criteria, such as the DNS hierarchy or IP address allocation, that could in theory be used to determine offline whether the binding is correct or not. In first-come-first-serve, there are no such external criteria and the authenticator itself allocates the names to its peers.

## 3 Security Considerations

The main advantage of LoF is that no third party is required. However, this implies a lack of knowledge for strong authentication. Without the LoF assumption, impersonation and/or MitM attacks are possible. Indeed, if the attacker appears in the first communication, he might claim the peer's identifier and give his own credential to the authenticator. Since the authenticator cannot distinguish an attacker from a honest peer, the attacker's binding would be naively accepted. Several factors exist that influence the likelihood of this incident, as follows.

**LoF scenarios.** In the design of LoF protocols, two components exist: the communication initiator and the LoF maker (authenticator). We devise a number of possible designs based on these components. In particular, two scenarios exist for the communication initiator, denoted as $Init = \{fixed, either\}$, that is, communication could be initiated by a fixed endpoint (e.g., in client-server model), or by either endpoint (e.g., in peer-to-peer model). Meanwhile, the LoF maker(s) could be one of the following: the communication initiator, the responder, a fixed endpoint, or both endpoints. We denote these as $LoFMaker = \{initiator, responder, fixed, both\}$. The Cartesian product $Init \times LoFMaker$ produces the set of LoF scenarios for our concern. Note that we may purge out the element $(fixed, fixed)$ as it is covered by $(fixed, initiator)$ and $(fixed, responder)$. For convenience, we denote each scenario by the initials of its components, e.g., **(FB)** stands for $(fixed, both)$.

The above two components could be used to identify potential attacks that might happen on each scenario, regardless of implementation details, purposes, or operating environments. As an example, in scenario (FI) where the LoF maker is the fixed initiator, the attacker must patiently wait for the first communication to take place before he could impersonate the responder. In contrast, scenario (FR) requires the responder to be the LoF maker, which allows an attacker to easily initiate the first communication and impersonate the honest initiator to the responder. Also in this scenario, MitM attacks are possible only if impersonation in the other direction could be done. This might be the case if the authentication in the opposite direction is insecure. In scenarios (FB) and (ER), however, MitM attacks are always possible because LoF is used bidirectionally. In overall, we summarize attacks on these scenarios as in Table 1.

**Table 1.** Attacks against scenarios in $Init \times LoFMaker$

| Scenario | Potential attacks |
|---|---|
| **(FI)** | Intercept communication attempt ⇒ Impersonate the responder; MitM attacks (if no strong authentication of the initiator) |
| **(FR)** | Intercept communication attempt ⇒ Impersonate the initiator; MitM attacks (if no strong authentication of the responder)<br>Initiate communication to the responder ⇒ Impersonate the initiator |
| **(FB)** | Intercept communication attempt ⇒ Impersonate both; MitM attacks<br>Initiate communication to the responder ⇒ Impersonate the initiator |
| **(EI)** | Intercept communication attempt ⇒ Impersonate the responder; MitM attacks (if no strong authentication of the initiator)<br>Initiate communication to both ⇒ MitM attacks (if no strong authentication of the initiator) |
| **(ER)** | Initiate communication to both ⇒ Impersonate both; MitM attacks |
| **(EF)** | Intercept communication attempt ⇒ Impersonate the initiator<br>Initiate communication to both ⇒ MitM attacks (if no strong authentication of the responder) |
| **(EB)** | Initiate communication to both ⇒ Impersonate both, MitM attacks |

**Binding multiple sessions.** LoF follows the *temporal separation* principle [10]: using some verification techniques, a communicating party can ensure that the peer it communicates with at time $t_1$ is the same as that at a previous time $t_0$. We call the period that such $t_1$ may fall in as the *authenticated period*, denoted by $[t_0, t_{end}]$. Applying to LoF principle, the first communication occurs at $t_0$, and subsequent communication should occur within $[t_0, t_{end}]$. Thus, to avoid future attacks, the design goal is to make sure that $t_{end} \to \infty$.

To achieve this goal, LoF implementation must be able to verify credential ownership, and that ownership must be unique. Otherwise, an attacker could claim the credential and bypass the authentication, even during $[t_0, t_{end}]$. Examples include plaintext exchange of shared key as credential, since the attacker can sniff the key and becomes its second owner. Moreover, only the authenticated peer should be allowed to continue communication, or else the attacker may intercept communication at a later time. For example, some systems allow the transmission of signed session key. Although signed by the peer's credential, it can be sniffed by the attacker and used to inject valid messages.

**Losses of bindings.** Another threat to temporal separation is losses of security bindings, due to storage failure, accidental deletion, re-installation, etc. If this happens at the LoF authenticator side, future communication appears to be the first communication, and thus the goal $t_{end} \to \infty$ fails. Moreover, events such as storage failure often lead to the loss of many bindings, which causes many first communication sessions to occur, thus increasing chances for attackers to successfully intercept at least one session. Also, if the peer lost its credential, it would have to create a new one, causing authentication failure. This may lead

to public announcement of the change to resolve failure, allowing attackers to attack at a specific time with higher chance to intercept a first communication.

**Authentication failures.** A failure occurs when the authenticator receives a credential that does not match the peer's binding locally stored. In many cases this implies an attack. For example, if the attacker appears only during the first communication and gives his own credential, then failures would occur on subsequent communication with the honest peer. Likewise, failures also happen if attacks are mounted on subsequent communication instead of the first one. Authentication failures may also be a result of losses of bindings. To resolve failures, the authenticator must detect which scenario is the cause. However, since the difference between these scenarios is subtle, the authenticator may either accept the new binding that allows the attacker to impersonate the honest peer, or deny the new binding of the honest peer and its legitimate communication.

**Attack environment.** Inherent characteristics of attack environments have certain influences on the attacker's success rate. In fact, there are several criteria that the attacker must meet in order to mount an attack, including his location, knowledge, and presence. If either route redirection or compromise of middle nodes is possible, the attacker could be anywhere in the network. Otherwise, he must be at either endpoint's local network (e.g., neighbours, university students), or on the route (e.g., ISP, intelligence agencies). The attacker might also need knowledge about an endpoint, such as its online time, or IP address that is not in the DNS and changes over time. The attacker's presence is also important for a successful attack, e.g., in MitM attacks, if an endpoint is mobile, the attacker must move along with it during the attack.

Another characteristic is the frequency of first communication sessions, which varies for each operating environment. The number of such sessions may be driven by the densities of authenticators and parties being authenticated, which is location-specific. It could also be influenced by the popularity of the LoF protocol. For example, an application-layer LoF protocol may be less used than a transport-layer LoF protocol. In addition, the dynamics of the community may as well affect the number of first communication sessions. As the community grows, or when there is churn [11], new pairs of peers and authenticators will be introduced, allowing attacks on the LoF bootstrapping process.

On the other hand, the risk of being detected is a deterrent to the attacker. This risk may be high given an effective attack detection mechanism. In certain situations, IDS techniques such as fingerprint-based misuse detection [12, 4.5] or anomaly-based detection [12, 4.4] could be used. To be feasible, these detection techniques should exploit the difference between an attacker and the honest peer, such as online time. Also, the honest peer and its authenticator may share some information through out-of-band channels, such as telephone, which the attacker is not aware of. These differences may result in the attacker's strange behaviors that make attack detection possible.

**Binding multiple protocol layers.** In some scenarios, the LoF protocol may be used as a transport layer carrying traffic for higher-layer applications. It is

possible that these layers are not aware of each other's operation. In that case, changes at the lower layer may not be visible to applications. If the attacker could change the peer's identifier and/or credential, the application would blindly continue the communication with the wrong endpoint. This happens, for example when the application and the LoF protocol use DNS names and IP addresses as identifiers, respectively. DNS spoofing would force the LoF protocol into using a wrong peer IP address and wrong security binding. Even with higher-layer authentication, this attack may still work if a mechanism such as plaintext password authentication is used, as password sniffing is possible.

**First communication detectability.** The ability to detect the first communication helps the attacker avoid being caught. When the attacker hijacks a local network with many nodes, he might encounter many LoF connections. Inherently, only a small fraction of these are first communication. Attacking all connections clearly reveals the attacker's presence, as most authentication sessions will fail. Instead, selective attacks against only first communications allow the attacker to stay stealthy.

Detectability may be facilitated by different factors. As an example, a LoF protocol maybe designed with different messaging for first versus subsequent communications. Also, the timing of first communications maybe leaked through out-of-band channels, such as emails, SMS messages, or by historical statistics. We devise in Figure 1 another detection method. First, the MitM attacker passes on messages in between until ($t_0$) when the authenticator successfully receives its peer's credential. He then waits until the authenticator responds ($t_1$), and check the waiting time $t_1 - t_0$ against $\epsilon$, the expected response time from the authenticator during a subsequent communication. If $t_1 - t_0 \gg \epsilon$, this is the first communication, as $t_1 - t_0$ is the time for the user to decide whether to accept the credential. Otherwise when $t_1 - t_0 \approx \epsilon$, no human decision is involved, and that may imply subsequent communication.

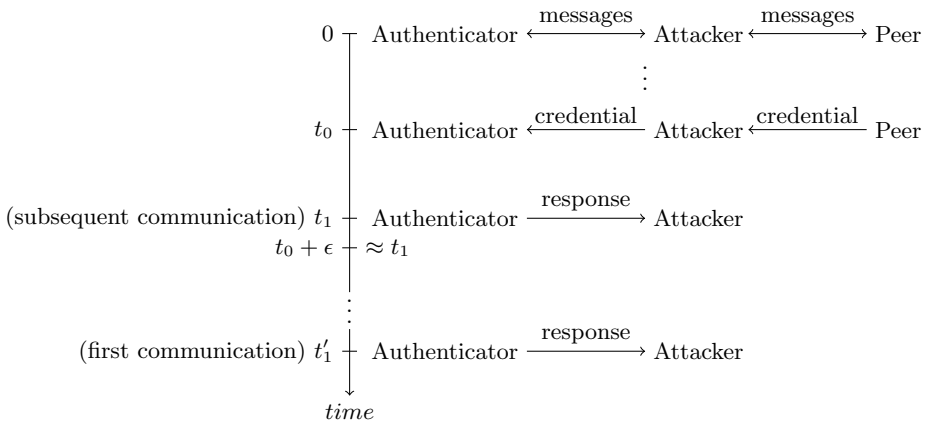


**Fig. 1.** Timing method to detect LoF first communication

The above attack makes use of two assumptions. Firstly, human being is involved in accepting/rejecting the credential. In fact, this applies to most LoF protocols that have been developed. Secondly, upon realization of the first communication, the attacker is able stop the authenticator from accepting the peer's credential and replace with that of his own. This attack exploits implementation defects rather than protocol issues, e.g., in certain WWW browsers as detailed in Section 5.

**Performance issues.** In most communication protocols, a communication state needs to be remembered by at least one communicating party. This party must store and perform lookup of state data in memory. This brings a chance for an attacker to create a massive number of sessions whose state data would either fill this party's memory or exhaust its capacity for managing the data. An early example is TCP, which suffers from SYN-flooding attacks [13]. LoF protocols are no exception, as the authenticator must store the peer's binding permanently and is thus vulnerable to resource exhaustion attacks.

## 4 Secure Shell (SSH)

In the early days, remote login protocols such as `telnet` and `rlogin` sendzz plaintext passwords over the networks where they could be sniffed. SSH [14] was created specifically as a response to prevalent sniffing attacks, using encrypted tunnels between the client and the server. SSH became popular because, thanks to LoF, it can be installed locally at the client and server and avoids most of the administrative overheads involved in strong server authentication.

The SSH server is identified by either its DNS name or IP address, with its public key being the credential. When the client initiates a connection, it receives the server's public key $PK$. The client looks in its storage of bindings, e.g., a `known_hosts` file, for a key associated with the server's identifier. If no key is found, the session is the LoF first communication, and the client user is asked whether to accept $PK$. If accepted, the binding from the server's identifier to $PK$ is stored. In case the server's key is already in `known_hosts`, the server is authenticated only if this stored key equals $PK$. Otherwise, the user would be alerted of a potential attack and, depending on the implementation, the session may be terminated.

SSH LoF has several characteristics that make it relatively secure. It follows scenario (FI): the communication is always initiated by the client, and it is also the only LoF authenticator. Although SSH is still vulnerable to MitM attacks, these are difficult to implement. The adversary must wait for the first communication to take place to attack successfully. Meanwhile, most casual hackers are unlikely to be on the communication route at the right time. More advanced attacks that reroute the traffic via the attacker are easier to detect with an IDS and will thus not last long. Mobile clients, on the other hand, have most likely initialized their `known_hosts` before roaming into insecure networks.

In another advantage, SSH is often comes as a single piece of software comprising both the security protocol and the terminal emulator. This eliminates

the problems caused by lack of application awareness to lower-layer protection: the SSH client takes care that the terminal sessions are consistently protected. The exception is with SSH port forwarding, in which case SSH carries traffic for other protocols such as SMTP, X11 or HTTP. Even then, the status of SSH is still controlled by the user with a separate interface, e.g., an SSH client.

One critical feature in the design of SSH is that a passive observer cannot distinguish between a first and a subsequent communication sessions. Thus, either the attacker ceases to attack, or he must target all connections and risks detection. In relatively static wired networks, the attacker may be able to spot new client computers or users and target them. However, the more vulnerable settings, such as open wireless networks, are actually difficult for the attacker because he cannot tell which clients are new. In such cases, an attack report policy would certainly help detect the attacker's presence. Also, the timing attack in Figure 1 would not work as its second assumption fails for most SSH implementations, such as OpenSSH and PuTTY. Upon user acceptance, these software force local storage of server's public key, without further concern. Unless the local program execution is intefered with, this storing process is unstoppable.

Despite above advantages, SSH also has some shortcomings. We recall that SSH sessions usually have two layers of authentication: LoF authentication of the server and password authentication of the client. The session key created in the key exchange will be only bound to the to server credentials but not to the user password or username. Hence, impersonation of the server is enough to learn the session key. Also, the use of plaintext password means that a successful attack against the LoF will also compromise the password. By using a challenge-response protocol for client authentication and by binding the session key to the client password these problems could be mitigated.

Another issue is that when the server private key is lost, the administrator would notify users in public. The short period following this announcement would be the best time for active attacks. The same problem may also emerge for environments with high and predictable churn rates, e.g., university networks. In such places, the attacker may know when new users appear in the network and target their first SSH logins.

## 5 Transport Layer Security (TLS)

Similar to SSH, TLS provides, among other things, authentication and confidentiality for peer-to-peer communication. While each peer is identified by its TLS certificate, authentication is accomplished with support from a PKI hierarchy of Certificate Authorities (CA). However, without such premises, LoF could be introduced as an alternative. While authentication in TLS can be peer-to-peer [15, F.1.1], we consider only its promient use nowadays, i.e., the client-server setting represented by scenario (FI), such as with HTTP. Thus, the similar security analysis of SSH regarding this scenario is also valid for TLS LoF.

Nevertheless, LoF TLS differs from SSH in other aspects. TLS is the transport-layer protocol, providing security services for upper layers. This implies its

greater popularity than SSH which is mainly used for remote logins. For instance, many WWW servers nowadays use TLS to provide security for HTTP, however without a proper TLS certificate issued by a trusted CA. Thus, users connecting to such servers via TLS would have to make leap-of-faith on the unrecognisable certificates. Due to the inherent popularity of WWW usage, there is a high chance for successful interception of a first communication session.

The fact that TLS is a transport-layer protocol also presents multiple-layer binding issues in which users are unaware of lower layer attacks on TLS LoF. This implementation issue appears, for instance, in popular software such as Firefox and Safari, making the attack in Figure 1 feasible. In particular, when Firefox receives an unverifiable server X.509 certificate, it displays a warning to the WWW user. However, to be able to examine the certificate and accept/reject it, the user must click on an **Add exception...** button, which triggers a reconnection to the server, again asking for its certificate. This latter certificate will be presented to the user instead of the first one. With the timing method, the attacker could detect the first communication, and is able to inject his own certificate during the reconnection, thus succeed in impersonating the server. Clearly, the difference between the two certificates is visible to TLS, but not to the user.

Another concern with current implementations of TLS, most notably WWW browsers, is the temporal separation principle, which signifies that a certificate, once accepted, should be stored permanently. However, Firefox and Safari give users an option, though non-default, to accept the TLS certificate for a particular session only. Even worse, this is the default option in Google Chrome, which implies that every connection is treated as the first communication, rendering the WWW user highly vulnerable to attacks.

## 6 Better Than Nothing Security (BTNS) IPsec

Unlike SSH, IPsec is designed for protecting all different types of network traffic. It operates at the Internet layer and establishes secure channels for IP packets, transparent to upper layers. The main operation of IPsec is with the Internet Key Exchange (IKE) protocol. Within this protocol [16, 1.2], public-key certificates are used to facilitate peer authentication. IKE is also used to exchange a shared keying material for establishing secure channels. The security of this material is bound to the public keys of both endpoints using digital signatures.

Normally, IKE requires signed public-key certificates or a Kerberos server (in the PKInit mode of IKE) to bind the public keys to the identifiers of the end nodes. Inherently, this kind of infrastructure does not span globally. BTNS mode of IPsec [17,18] is a proposed extenstion of IPsec that supports LoF. Its goal is to perform anonymous encryption of communication and authentication within one communications session. Since the specification of BTNS is still undergoing, we have to make some assumptions in our discussion.

**LoF scenarios.** BTNS LoF may first appear to be similar to SSH. If used below the `telnet` application, it provides comparable security to SSH. With fixed client

and server roles, it follows scenario (FI). However, generic IPsec architecture does not follow this kind of asymmetric thinking. Instead, BTNS seem to be specified with scenario (EB) in mind where either endpoint can initiate the communication and both make leaps of faith on each other. This allows the attacker to initiate a first communication to any other node, causing that node to bind any previously unbound name to the attacker's credential. This way, the attacker can hijack a server name at a client's namespace before the client connects to the server.

As a remedy, BTNS could possibly be implemented in a more restricted way. For example, the binding could be created only at the initiator, as in scenario (EI). It may first appear that this will prevent the reversal of roles and teaching the other party a false binding. However, in IPsec it is very difficult to be certain which endpoint actually initiated the communication. This is because IPsec may be triggered by another communication in opposite direction. A typical example is FTP where the server initiates connections to the client. If only the FTP data transfer is protected by BTNS IPsec, then the LoF initiator will appear to be the server. Thus, an attacker can cause the FTP server to initiate a first communication to his IP address, making attacks significantly more likely.

Similar reversals of IPsec initiator and responder roles occur in Windows implementations of IPsec where a client workstation can be configured to be in responder-only mode, i.e., to use IPsec only if the server initiates connections to it. The role reversals are perfectly acceptable in IPsec with strong authentication and fit well the symmetric design of the IPsec architecture. Authentication with LoF, however, does not fit well into this setting. This is easiest to understand in peer-to-peer protocols where there is no difference between client and server, and scenario (FI) with fixed initiator is simply nonsensical.

**Binding of protocol layers.** As IPsec operates at IP layer, it introduces a operation awareness problem to higher layers, causing a number of attacks [19], especially when BTNS LoF is in use. One approach to resolve this is called *connection latching* [20], which binds each upper-layer connection (e.g., TCP) to the underlying IPsec channel protecting it. This ensures that changes in the underlying IPsec channel would terminate the upper-layer connection. More importantly, an interface is provided for upper layers to monitor IPsec connections. Another approach is *channel binding* [21], which works when the higher-layer application provides strong authentication of the peer. In that case, this strong authentication is used to securely verify the public key received via BTNS. Both mechanisms solve many of the security problems that we later identify in HIP, such as the lack of binding among protocol layers.

Apart from host-to-host connections, IPsec can also be used to create tunnels between security gateways or between a host and a gateway. These are actually more common IPsec scenarios because they make up VPN tunnels. If BTNS is used, IPsec and the application-layer connections are implemented on different machines, making inter-layer binding methods such as connection latching and channel binding infeasible. Given that BTNS IPsec have yet been deployed, it is impossible to point out actual security failures. However, the possibility of

LoF-authenticated IPsec association expiring underneath a long-lived application session is an important concern when specifying BTNS-based VPNs.

**Managing security bindings.** The intended transparency of IPsec to application layers brings further issues when BTNS is used. Being blinded from IPsec operation, user confirmation on the correctness of security binding and offline recovery of authentication failures is typically impossible. Moreover, even if an IPsec-awareness interface is implemented on end hosts, a user behind an IPsec gateway will not be able to control the gateway and will not even know what is causing the communication failure if the gateway detects change of peer key and refuses to route further packets.

Secondly, an attacker can flood BTNS IPsec by repeating first communication sessions between a target host or gateway and presenting each time a different identifier and credential. The target will have to decide which bindings to store and which to drop, but it has not feedback from the application layer on which bindings correspond to application-layer sessions that still exist.

Thirdly, IPsec host mobility maybe problematic if BTNS authenticator, as intended, stores a binding between the peer IP address and its credential (i.e. public key). When the same IP address is reused by a different mobile device, the authenticator will see that as an attack. Similarly, hosts behind a NAT share the same IP address but have different credentials, which again will look like an attack. This makes IP addresses unsuitable identifiers for BTNS bindings.

## 7 Host Identity Protocol (HIP)

HIP is similar to IPsec in that it provides network authentication transparently to higher-layer applications. HIP also aims at mobility and multi-homing. To do so, it inserts a *host identity layer* into the TCP/IP model, between the IP and the transport layers. On this layer, each node is identified by a *Host Identifier* (HI), which is actually a public key. This allows each HI to be self-certifying, which avoids impersonation. Also, HIs replace IP addresses as communication identifiers for upper-layer protocols, e.g., TCP. As a result, these protocols become independent of IP address changes, thus supporting mobility and multi-homing.

The core of HIP is the base exchange [22]. It involves four messages, as shown in Figure 2. Message I1 is used by the HIP initiator to trigger the connection. It contains the hash of the responder's HI, called Host Identity Tag (HIT). Then, R1 and I2 are used to exchange the Diffie-Hellman key for the secure channel. Also, the initiator includes in I2 its HI to allow the responder to authenticate its messages. Finally, the message R2 completes the exchange.

Using HIs as identifiers, the problems mentioned in the previous section about dynamic and non-unique IP addresses are irrelevant in HIP because the IP address is only used for routing and not for security bindings. However, the problem is at the application layer where DNS names are typically used as identifiers. In this case, secure mapping from DNS names to HIs is needed. This should be solved by a secure name resolution mechanisms such as DNSSec [23]. The initiator would look up the responder HIT in DNS and send its own DNS name in

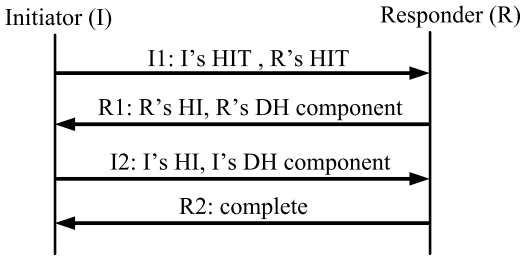


**Fig. 2.** A simplified operation of HIP

the I2 message, which the responder can verify from DNS. If both sides perform the lookup from secure DNS, then strong authentication is achieved.

There are several difficulties with the use DNSSec in HIP. Firstly, while DNSSec is being deployed, it still does not cover all Internet individual hosts. Secondly, not all DNS servers support HIs in DNS records. Thirdly, IP address lookup mechanisms may not necessarily support indirect mappings from DNS name to HI to IP address. This may leave a HIP host with just the knowledge of the peer's IP address rather than HI. These are the situations where LoF becomes an attractive option. The *opportunistic mode* (LoF) [24] in HIP works as follows: the initiator sends a tentative message I1 to the responder, which—if it supports HIP—responds with R1 and includes its DNS name (or other application-layer identifier) as well as its HI in the message. The initiator continues by revealing its own name in I2. Both may then store a mapping from the name to the HI.

**Exploiting the Symmetric Operation.** The LoF protocol described above corresponds to scenario (EB), which enables the attacker to connect to any hosts and teach them false bindings. The initiator may also choose to be anonymous, in which case the responder stores no binding. If this was taken as the only allowed mode of operation, then it would correspond to scenario (EI), which is slightly less vulnerable. There will nevertheless be the possibility of role reversal attack similar to those against BTNS because of the symmetric nature of HIP.

**LoF Bootstrapping Detection.** Unlike the messaging design in SSH, the HIP base exchange allows the attacker to detect the bootstrapping process (first communication). Specifically, when the initiator does not know the responder HI, it sends the I1 message with a null responder HIT. Seeing this null value, the attacker can be certain that the first communication is happening. On the other hand, with regard to the timing method in Figure 1, current implementations of HIP such as OpenHIP and HIPL are invulnerable, since the acceptance of credentials is an automatic process, i.e., it does not involve human decisions.

**Problems of Transparency.** HIP is designed as a general protocol operating in the middle of the TCP/IP model. Its implementation is independent of higher-layer applications. Also, HIP provides an interface for these applications

to communicate across networks. Many legacy applications do not understand HIP, as well as its protection. This is the main source of problems for HIP.

In HIP, authentication of the responder is simple. The initiator originates a communication and specifies which HI it wishes to communicate with. If the responder uses a different HI, the connection will fail. Otherwise, only the owner of the HI can successfully establish the HIP connection. Conversely, the responder may not always authenticate the initiator. For example, if the initiator does not provide its DNS name in the base exchange, it cannot be authenticated, and the responder is left with an anonymous peer. Also, providing a name that cannot be resolved achieves similar effect. The problem is, upper-layer applications are not aware of this fact, and they may unintentionally use this connection. This is mainly due to bad implementation and configuration of HIP.

**Weak HI-to-LSI Mappings.** A problem with HIP is that legacy applications do not understand HI or HIT. HIP solves this by using Local Scope Identifiers (LSIs). Depending on the application, an LSI could be in a form of an IPv4 or IPv6 address. When the application wishes to communicate with a DNS name, HIP returns an LSI as the result of a DNS query. This LSI locally represents the actual peer, and is thus mapped to the peer's IP address and (if available) HI. When a connection is made to this LSI, HIP searches for a mapping of this LSI, and performs the base exchange with the corresponding peer.

There are currently two main methods for LSI-to-HI mapping. The first method uses a mapping list to perform lookup. When a new HI is recognized, HIP picks a free LSI from its local pool and maps the two together. Since the mapping list is bijective, authentication of LSIs is secure. In the second method, the mapping is such that the LSI is the value of a function $f$ on the HI. If $f$ were collision-resistant, then each LSI is a self-certifying. However, it is problematic that the range of LSIs is small. For example, a private IPv4 range such as 10.0.0.0/24 may be used as LSIs. There are only $2^{24}$, or approximately 16 millions possible LSIs, and it is easy to find two HIs that map to the same LSI. Consequently, the attacker may trick the application into communicating with him instead of the honest peer, since they are represented using the same LSI.

The above weaknesses open a number of attacks on HIP. In particular, we consider impersonation attacks in a peer-to-peer model between Alice and Bob, given that they have already bootstrapped LoF. We also assume that the HI-to-LSI mapping is done using a function $f$ instead of a mapping list. The attacker starts by computing a HI that would map to the same LSI as Bob's HI would. Then, he establishes a HIP connection to Alice, using this new HI, which results in Alice mapping this HI to the LSI she used previously for representing Bob. When Alice's higher-layer application reconnects to Bob through such LSI, HIP does not establish a new HIP connection to Bob, but simply forwards the communication to the attacker's HIP connection, making the attack successful.

In another attack variant, it may be possible to replace the underlying security association in the middle of a higher-layer session. In some implementations (e.g., OpenHIP), the HIP security association is discarded after an idle period. However, the higher-layer communication has not necessarily terminated, e.g., it

could happen during long TCP idle periods. When TCP becomes active again, the HIP base exchange would again take place and the LSI would be rebound to a different HI, e.g., using the above attack. Because the upper-layer protocol is unaware of this change, an attacker may take over the connection. Also, since the attacker hijacks the upper-layer connection in the middle of it, he avoids any initial application-layer authentication that may have been initially required.

**Exhausting LSI Namespace.** In some HIP implementation, such as OpenHIP, acceptance of security bindings is automatic, without user confirmation. This allows potential exhaustion of the LSI space, which happens when the LoF maker is the HIP responder or the attacker manages to reverse the roles of LoF initiator and responder by triggering the target to be the initiator. In particular, the attacker may create many HIs, and uses each of them to communicate with a particular HIP responder. If this node uses a HI-to-LSI mapping list, there are two problems. As the attacker continues to generate HIs, the LSI space at the targeting HIP node will be filled up, resulting in dropping/rejecting of legitimate bindings. Even if the LSI space is large enough (e.g., IPv6-like LSI), it is questionable on how HIP manages such a large set of bindings efficiently.

## 8 Strengthening LoF Protocols

In previous sections, a security analysis shows that LoF brings certain security to protocols for which it is applied. However, these protocols still have some weaknesses that may lead to attacks, e.g., impersonation and MitM. The chance that these attacks occur also varies from protocol to protocol, and environment to environment. We now present several proposals that address the revealed weaknesses and thus may strengthen the overall security of LoF protocols.

### 8.1 Multi-path Authentication

To cope with MitM attacks, [25] introduce an idea that statistically improves the ability of detecting whether attacks are really happening. This idea is similar to the requirements on *vertex disjoint paths* between two nodes. Basically, communication between Alice and Bob is secure if there are $n$ communication paths between them, and there are $t$ attackers such that $n > 2t + 1$ [26, III].

The use of multiple paths could as well be applied to LoF protocols. The main concern is the requirement of disjoint paths, which maybe problematic because most network nodes have only one Internet connection. To simulate disjoint paths, Alice needs secure communication to her friends, e.g., Carol and Dave, each of which has a distinct connection to Bob. As an example, in SSH, Carol and Dave could be SSH servers whose credentials are already in Alice's `known_hosts`. Unlike in SSH, in other protocols such as TLS, HIP and BTNS, there might exist public infrastructures whose security bindings could be securely verified, such as public rendezvous servers in HIP, or small PKI hierarchies. These could as well be used as Carol and Dave.

### 8.2 Resolving Authentication Failures

As previously mentioned, the main problem with authentication failures is the difficulty in distinguishing their causes. In this section, we propose approaches that could be used to resolve failures for some specific situations.

Firstly, we distinguish two scenarios in which attacks appear during LoF first and subsequent communication, respectively. We notice that most attackers are not always active, whereas in situations like the client-server model, the server is always online. This gives a possible method: after the first communication, the LoF authenticator (e.g. the client) keeps probing the peer (e.g. the server) for its credential. These probes should be made at random times. Since the attacker cannot guess when these happen, he cannot intercept all probes. This method is especially effective with a mobile peer, because the attacker has to move along with this peer to execute attacks. If the peer always answers with the same credential, it is unlikely to be an attacker.

The second approach deals with the confusion in authentication failures caused by either loss of private keys or attacks. To resolve this confusion, the system administrator may provide information through out-of-band channels to users. For example, consider a SSH server that lost its private key and is assigned a new one. In this case, the administrator could tell a secret word to the users over the phone and the SSH server could require the client to send this word encrypted by the new public key. That way, the users are forced to contact the administrator before they can reconnect to the server, and they cannot just click ok to accept the new public key.

### 8.3 Best Practices for LoF Applications

Together the discussion of security considerations and the analysis of protocols reveal several principles that a LoF application should satisfy. These best practices are important for protocol designers to consider when applying LoF into their protocols. Also, they may be helpful during protocol implementation and deployment. These principles are summarized in the following list:

- To prevent attacks on LoF first communication, the LoF responder should not make any leap of faith. Note that care must be taken in determining the actual initiator, for example when one endpoint uses an external protocol to trigger LoF from the other endpoint.
- The decision to accept a security binding must be based on user confirmation. This allows out-of-band information to support verification of the binding. Also, it avoids flooding the LoF authenticator with security bindings.
- To facilitate attack detection, a credential accepted for communication must be permanently stored, or removable only by advanced users.
- Within a communication session, the secure data channel must be bound to all authentication processes. All credentials of both endpoints (e.g., keys, passwords) must be involved in creating the session key. Thus, the secure channel can only be compromised if all the involved authentication mechanisms fail.

- For application-independent LoF implementation, there should be an interface (e.g., APIs) for applications to monitor its operation. Failures to do so lead to exploitation of the application's oblivion to changes such as session restarts or policy changes.
- The LoF first communication must be made indistinguishable from the subsequent ones to sniffing attackers. This forces attackers to perform attacks on all connections, hence risking detection.
- Where possible, authentication over multiple paths should be used during LoF bootstrapping or authentication failures, which defeats MitM attackers that are not on all these paths.
- Attackers can sometimes predict the timing of the LoF first communication, e.g., after storage failures. In such cases, out-of-band channels (e.g., telephone, SMS) maybe needed to support the verification of the peer's credential.

## 9 Conclusion

In this paper we study the security of the LoF mechanism as it is applied to various network protocols. In principle, LoF allows the authenticator to simply accept the peer's binding during the first communication between them. This binding is used to authenticate the same peer for later communication. The security of this mechanism relies on the assumption that the LoF bootstrapping process (first communication) is attacker-free.

We investigate why this rather weak authentication mechanism has been successful in SSH. To do so, we consider different aspect of LoF protocols such as the choice of initiator and LoF authenticator, temporal separation, binding of multiple protocol layers, and performance issues. The environment where attacks might occur is also taken into consideration. Using these aspects, we pointed out some critical features that often shield SSH against attacks on the LoF first communication. We also apply the same analysis to TLS and some other protocols for which LoF has been proposed: BTNS IPsec, HIP. It turns out that the security of LoF authentication in these protocols is not comparable to SSH and that it depends heavily on the way the protocols are implemented and used.

The analysis reveals various attacks on these LoF protocols, including impersonation, MitM attacks, and flooding of credentials. We then propose several mechanisms to help reducing the success probability of these attacks, including best-practice guidelines for the design of LoF protocols that should be considered for this weak authentication to be reasonably secure.

**Acknowledgements.** This work was supported in part by the Academy of Finland (project no. 135230). Also, the authors thank Carlos Cid and the anonymous reviewers for helpful comments.

## References


1. Kohl, J.T., Neuman, B.C., Ts'o, T.Y.: The Evolution of the Kerberos Authentication Service, pp. 78–94. IEEE Computer Society Press (1994)
2. VeriSign, Inc.: VeriSign Certification Practice Statement (2009), http://www.verisign.com/repository/CPS/
3. Potter, B.: Dangerous URLs: Unicode & IDN (2005), http://www.sciencedirect.com/science/article/B6VJG-4FVC3YD-6/2/9d0fa84d322964a8c9ac42cba2936dea
4. Abdul-Rahman, A.: The PGP Trust Model. The Journal of Electronic Commerce 10(3), 27–31 (1997)
5. Jsang, A.: An Algebra for Assessing Trust in Certification Chains. In: Network and Distributed Systems Security Symposium (NDSS 1999), San Diego, USA (1999)
6. Arkko, J. (ed.), Kempf, J., Zill, B., Nikander, P.: SEcure Neighbor Discovery (SEND). RFC 3971 (2005)
7. Aura, T.: Cryptographically Generated Addresses (CGA). RFC 3972 (2005)
8. Aura, T.: Cryptographically Generated Addresses (CGA). In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 29–43. Springer, Heidelberg (2003)
9. Baek, J., Newmarch, J., Safavi-naini, R., Susilo, W.: A Survey of Identity-Based Cryptography. In: Proc. of Australian Unix Users Group Annual Conference, pp. 95–102 (2004)
10. Arkko, J., Nikander, P.: Weak Authentication: How to Authenticate Unknown Principals without Trusted Parties. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2002. LNCS, vol. 2845, pp. 5–19. Springer, Heidelberg (2004)
11. Stutzbach, D., Rejaie, R.: Towards a Better Understanding of Churn in Peer-to-Peer Networks. Department of Computer Science, University of Oregon (2004)
12. Mchugh, J.: Intrusion and Intrusion Detection. International Journal of Information Security 1, 14–35 (2001)
13. Eddy, W.: TCP SYN Flooding Attacks and Common Mitigations. RFC 4987 (2007), http://tools.ietf.org/html/rfc4987
14. Ylonen, T.: SSH - Secure Login Connections over the Internet. In: Proceedings of the 6th USENIX Security Symposium, pp. 37–42 (1996)
15. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (2008), http://tools.ietf.org/html/rfc5246
16. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol. RFC 4306 (2005), http://tools.ietf.org/html/rfc4306
17. Williams, N., Richardson, M.: Better-Than-Nothing Security: An Unauthenticated Mode of IPsec. RFC 5386 (2008)
18. Touch, J., Black, D., Wang, Y.: Problem and Applicability Statement for Better-Than-Nothing Security (BTNS). RFC 5387 (2008)
19. Aura, T., Roe, M., Mohammed, A.: Experiences with Host-to-Host IPsec. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2005. LNCS, vol. 4631, pp. 3–22. Springer, Heidelberg (2007)
20. Williams, N.: IPsec Channels: Connection Latching. Internet Drafts (2005), http://www.ietf.org/id/draft-ietf-btns-connection-latching-11.txt
21. Williams, N.: On the Use of Channel Bindings to Secure Channels. RFC 5056 (2007), http://tools.ietf.org/html/rfc5056
22. Moskowitz, R., Nikander, P., Jokela, P. (ed.), Henderson, T.: Host Identity Protocol. RFC 5201 (2008), http://www.ietf.org/rfc/rfc5201.txt

23. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. RFC 4033 (2007)
24. Komu, M., Lindqvist, J.: Leap-of-Faith Security is Enough for IP Mobility. In: Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference, CCNC (2009)
25. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In: Proceedings of the USENIX Annual Technical Conference, Usenix ATC (2008)
26. Desmedt, Y.: Unconditionally Private and Reliable Communication in an Untrusted Network. In: IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security, pp. 38–41 (2005)