Identity Certified Authentication for Ad-hoc Networks

André Weimerskirch
Communication Security Group
Ruhr-University Bochum
44780 Bochum, Germany
weika@crypto.rub.de

Dirk Westhoff
Network Laboratories
NEC Europe Ltd.
69115 Heidelberg, Germany
dirk.westhoff@ccrle.nec.de

ABSTRACT
Ad-hoc networks face huge security lacks. In the most general case entities need to build up a well-defined security association without any pre-established secret or common security infrastructure. In previous work we presented a protocol which provides a weak form of authentication that we call zero common-knowledge (ZCK) authentication. The protocol is extremely efficient and only requires symmetric primitives but does not provide identification. In this work we extend this approach in such a way that our new protocol provides identification at the cost of external infrastructure and moderate computing power. Our new protocol can be used to authenticate messages, e.g., to exchange keys for the earlier ZCK authentication protocol. Compared to public-key schemes, our approach is still very efficient.

Categories and Subject Descriptors
C.2 [**Computer-Communication Networks**]: General—*Security and protection*; E.3 [**Data Encryption**]: Public key cryptosystems

General Terms
Security, Algorithms

Keywords
Ad-hoc Networks, Security, Authentication, Identification, Key-chains

1. INTRODUCTION
Wireless ad-hoc networks face huge security lacks. In the most general case entities need to be able to establish well-defined security associations without any pre-established secret or common security infrastructure. We showed in [6] how to establish a security association in the general case by means of *zero common-knowledge (ZCK) authentication* in a very efficient manner by key-chains. We first recall the security objective of ZCK authentication: A is able to authenticate B in a zero common-knowledge fashion if A is able to identify again the authority that runs B, i.e., B is able to convince A that both had some relationship in the past. We also say that A *recognizes* B. Such a definition makes sense since unlike in military or closed networks where there is a single logical and commonly agreed trust authority we cannot assume such a situation in the general case.

Above approach aims at sensor and ad-hoc networks without any kind of infrastructure. However, it lacks the possibility of establishing a trust relationship based on the identity (which probably is not possible to provide at all without any infrastructure or previous knowledge). In this work we assume that there are an infrastructure and devices with moderate computing power and some storage resources to provide identity authentication. Hence it is clear that here our scope is limited to ad-hoc networks but no sensor networks. We use the term identification and identity certified authentication in the sense that an entity A is able to prove its identity to an entity B assuming that both entities trust some third party.

We especially foresee two scenarios. In the first scenario there is an ad-hoc network of portable devices which are connected to the Internet via a base station. We call such a network an *ad-hoc stub network*. Base stations are already installed at airports and hotel lounges and might be extended in the near future to fit such a scenario. The second scenario are ad-hoc networks without a permanent connection to the Internet or a base station but with a temporary connection. The mobile devices are connected to a PC once in a while, e.g., to charge the battery or synchronize data with the PC. We call such a setting *temporary infrastructure ad-hoc network*.

In case that the ad-hoc network consists of powerful devices, it is possible to deploy a traditional PKI scheme. In such a case each device holds a set of certificates of the certificate authorities (CA), and a certificate issued by a CA. For each signature or key exchange operation a device now has to perform a signature generation as well as a verification. The certificates authenticate the public key of an entity and thus prove its identity. If the certificates are issued with short life-time such that certificate revocation lists (CRL) are unnecessary, there is neither a permanent connection to a CA directory nor frequent CRL updates necessary. If a permanent on-line connection to a trusted third party (TTP) is available, it is also possible to deploy a purely symmetric system. In such a model each device shares a secret symmetric key with the TTP. The secret key might come with the device, or it might be exchanged on a secure channel such as short-range infrared or by a physical connection. To establish a trust relationship between devices A and B, A sends a request to the TTP which then chooses a session key at random and sends it to A and B, encrypted with the shared secrets of A and B, respectively.

We present a third approach for proving identity in this work which does not require a permanent connection to a TTP nor extended use of digital signatures. We assume that there is a temporary Internet connection available as it is in both the scenarios that we foresee. It should suffice to establish a connection to a server rarely, e.g., once a day. For instance, a server connection is available when the device's battery is charged and put in a cradle connected to a PC which in turn is connected to the Internet. We furthermore assume that the devices have moderate computing power and storage resources such as a simple PDA worth less than a \$100. These devices are not capable of performing frequent public-key operations though. We require that the devices are loosely time synchronized to a global time which can easily be done when a server connection is established. We do not assume anything else such as tamper resistance or a build-in unique ID. We call such devices *moderate power devices*. Our approach is an extension of the ZCK authentication protocol. It only requires one signature verification to establish a trust relationship between two moderate power devices. The protocol allows two entities that do not share any pre-knowledge to identify each other in a mutual way. It furthermore provides the exchange of a key-chain's final element which in turn can be used in our original protocol for all following identification and message authentication processes. Our new protocol can therefore be seen to be equivalent to the certificate exchange of a public-key scheme. After the certificates are exchanged, only our original authentication process which is extremely efficient needs to be performed for identification and message authentication. We call our new scheme *identity certified (IC) authentication*.

The main contribution of this work is an identification scheme suited to ad-hoc networks which shortens the gap to digital signatures whereby preserving the high efficiency of symmetric schemes. The scheme is still not as powerful as digital signatures since it only provides mutual identification and no signature capabilities[1] but we show that there is use for it in a variety of applications. Thus we analyze in which scenarios the original ZCK authentication protocol can be used, i.e., in which environments it is practicable and efficient to use the protocol as a building block. The paper is organized as follows. Section 2 gives a description of previous work including the ZCK authentication protocol, and presents scenarios suited to ZCK authentication. Section 3 introduces our new scheme. Section 4 compares our new scheme to public-key and symmetric solutions, and rates them by comparing their complexity and by giving advice for which scope they are applicable. Finally, the last section concludes this work.

[1]A signature can be checked by multiple entities whereas we only provide a mutual scheme. Hence our scheme is closer to a MAC scheme for which the key-exchange is done mainly with symmetric primitives.

2. RELATED WORK

We first recall the ZCK authentication protocol of [6] and then describe how it can be used as a building block to secure protocols in an ad-hoc scenario.

2.1 ZCK Authentication

Consider the case where Bob wants to authenticate to Alice. Let x_B^G be a t-bit string which is Bob's global private key. Let each device have a network identifier ID,[2] in our case ID_A and ID_B. Let f be a function that maps the identity ID to a bit-string, r_{ID} be t-bit random strings, and $\oplus$ be an operation on bit strings. Then x_0^B is Bob's private key for the communication with Alice, $x_0^B = x_B^G \oplus f(ID_A) \oplus r_B$, and $x_0^A = x_A^G \oplus f(ID_B) \oplus r_A$ is Alice's private key for the communication with Bob. Note that these keys are only applicable to the communication pair Alice and Bob, and to no one else.

We define a hash-chain, which is also known as Lamport's hash-chain [4], as $h(x_i) = x_{i+1}$ with x_0 being the anchor and h being an unkeyed one-way hash function that has a security of t-bits, i.e., which maps bit-strings to t bits. Our scheme is based on such hash-chains with x_0^A and x_0^B being the anchors. Let $x_{n_A}^A$ and $x_{n_B}^B$ be the final elements of the hash-chains, respectively. We call these the public keys of Alice and Bob. We can use a key-value of the chain to generate an authenticated message by a MAC (keyed hash function). Let $(m)_x$ be the MAC of a message m by the key x. The core idea of the protocol is as follows: First exchange a value $f(x)$ which the receiver will tie together with some experience. Then prove knowledge of the pre-image of $f(x)$, i.e. x, in order to authenticate by establishing a relationship to $f(x)$ and the past experience. In order to repeat the authentication step arbitrary many times a key-chain based on a one-way hash function is used. The protocol works as follows:

[2]For the protocol this is an identifier to recognize another entity again. We do neither assume that the ID cannot be tampered with nor that IDs are unique. A just has to be able to map B to some name ID_B.

```
1 : $A$ sends her public key $x^A_{n_A}$ to $B$, who
    stores $(x^A_{n_A}, r_B, n_B, 1)$
2 : $B$ sends his public key $x^B_{n_B}$ to $A$, who
    stores $(x^B_{n_B}, r_A, n_A, 1)$
Repeat Steps 3 to 9 for each authentication
    process
3 : Assume $A$ stores $(x^B_i, r_A, j, u)$ and
    $B$ stores $(x^A_j, r_B, i, v)$
4 : (+) $A$ sends authenticated messages
    $(m)_{x^A_{j-u-1}}$ to $B$
5 : (+) $B$ sends authenticated messages
    $(m)_{x^B_{i-v}}$ to $A$
6 : $A$ opens her key by sending $x^A_{j-1}$ to $B$.
7 : $B$ checks if $h(x^A_{j-1}) = x^A_j$
8 : For $k = 1$ to $k' \leftarrow \max\{u, v\}$ repeat Steps 8.1
    to 8.5
  8.1 : $B$ opens his key by sending $x^B_{i-k}$ to $A$
  8.2 : $A$ checks if $h(x^B_{i-k}) = x^B_{i-k+1}$
  8.3 : $A$ opens her key by sending $x^A_{j-k-1}$ to $B$
  8.4 : $B$ checks if $h(x^A_{j-k-1}) = x^A_{j-k}$
  8.5 : If any check fails, or the loop is
        interrupted, $A$ and $B$ stop execution.
        Then $A$ stores $(x^B_i, r_A, j, \max\{u, k+1\})$
        and $B$ stores $(x^A_j, r_B, i, \max\{v, k+1\})$.
9 : $A$ and $B$ store the new values
    $(x^B_{i-k'}, r_A, j-k'-1, 1)$ and $(x^A_{j-k'-1}, r_B, i-k', 1)$
```

Note that for above protocol Alice and Bob always have to be in the same role, i.e., Alice always plays the part of A and opens her key first (Step 6). The order becomes regardless by adding another key-opening step, i.e., by requiring the opening of two keys on each side. Furthermore it is possible to update a key chain, i.e., to use a new chain once the old one has no elements left without loosing the trust association. Steps 4 and 5 are optional and provide message authentication. In that case it has to be ensured that all messages were received before continuing with Step 6. This can be done by a flag that is also protected by the MAC indicating the last data packet of the message.

The authentication scheme is secure against a passive eavesdropper, and for message authentication it is even secure against an active adversary [6]. It is widely believed that computing a collision to a given message (target collision resistance) in a one-way hash function that maps strings to t-bits, with $t = 80$, is as hard as factoring an RSA modulus of 1024-bits. We believe that target collision resistance is sufficient for our application whereas collision resistance (finding any pair of colliding messages) requires twice the number of bits. Hence to establish a security level similar to 1024-bit RSA we assume $t = 80$. Let the size of the average key-chain be $n = 100$ which should meet the demands of most short-lived pervasive and ad-hoc networks. As we said before authentication in an ad-hoc network is usually connected to some service that is offered or requested, i.e., authentication is bound to messages that are exchanged. Altogether the scheme requires on average 50 applications of the hash function, and three message exchanges each of 10 bytes. It furthermore requires 24 bytes of storage on the prover's and verifier's side which are far less than an RSA public key and about the same size of a 160-bit ECC public key. Note that our scheme ensures mutual authentication such that the prover and the verifier have to store the other entity's public key. Since most relationships in a ad-hoc network are mutual we do not believe this to be a disadvantage but a feature. A hash function has very low running time. For example, the SHA-1 implementation in [1] obtains hashing speeds of 48.7 Mbit/s whereas an RSA verification (which is one of the most efficient signature verification scheme) runs in roughly 1 ms [2]. Thus an application of a hash function is faster than an RSA verification by almost a factor of 10^6, and the repeated application of a hash function nearly is for free for 50 or even more iterations. If a device has on average contacts to $p = 50$ communication partners, there is storage needed of 1200 bytes.

2.2 ZCK as Building Block

In this section we consider existing protocols and propose to use ZCK as a building block for these. The first class we consider are routing protocols such as distance vector routing which is widely used in IP networks and can also be used in wireless multi-hop networks where each node acts as a router. Each router maintains a routing table listing all possible destinations in the network. For each entry of the routing table there is the information of the next router to the destination on the shortest path assigned. When routing a packet to the destination, the node sends the packet to the next router as determined by the routing table. The routing tables are maintained by periodical updates. Each node then transmits an update to its neighbors. The most powerful attacks that are known are the blackhole attack and the routing loop attack. First attack aims to route all data via an attacker's router by advertising short distances to all destinations via this router. The second attack injects routing loops such that a packet is sent along that loop several times.

These update messages can be secured by ZCK authentication for several reasons. First, the update information does not need to be encrypted but only needs to be authenticated to prevent nodes of manipulation the routing entries. Assuming that all messages in the network are authenticated using ZCK message authentication the blackhole attack degenerates to a denial-of-service attack. Furthermore, the nodes are mobile such that a node's neighborhood often changes and efficient mechanisms are needed. It is clear that above ZCK authentication protocol cannot prevent all possible attacks. Denial-of-service attacks are always hard to prevent, especially if there is no identification of the nodes possible. However, we believe that even without an infrastructure, it is possible based on experience and heuristics to build a robust routing scheme. For example, a node might only accept routing updates of neighbor nodes of which he received another service yet. A node could also rate update messages according to the traffic he has to forward in the next few minutes. In the following time he could only accept those update messages of nodes below a certain threshold. This will prevent the routing loop attack. The IC authentication which we introduce later gets rid of these limitations such that it can be used to secure the maintenance messages in a strong sense. The authenticated key-chains which can be exchanged during the IC authentication might then also be used in combination with secure routing mechanisms using key-chains as proposed in [3].

Another class of application we expect ZCK authentication as well as the IC authentication suitable for are peer-to-peer distributed systems. In such a system a large number of nodes can potentially be pooled together to share their resources, information and services. To make this vision become reality in ad-hoc networks with restricted devices, lightweight authentication schemes are mandatory. Depending on the network architecture, which may be either with or without temporary connection to the fixed network, we propose to choose the ZCK or the IC authentication. Both schemes ensure a bilateral authentication verifiable by the communication partners Alice and Bob. Next, we describe the IC authentication in detail.

3. IDENTITY CERTIFIED AUTHENTICATION

In this section we extend our previous scheme for the most general case to provide authentication and identification. We assume that the devices are able to perform a signature verification which is very reasonable in the case of RSA with short public exponents and only requires moderate computing power and memory storage, and that devices are loosely time synchronized to a global time. We also assume that the device has access to a temporary connection to a server which performs pre-computations. Our goal here is to extend the ZCK authentication scheme for the general case in such a way that it provides identification to entities that do not know each other nor had any contact in the past, i.e., such that an entity knows the unique identification name of the entity it is communicating with. For that reason we suggest a protocol that provides mutual proof of the identity of two entities. This can be seen similar as the exchange of certificates in the public-key scenario. Once the identity proof was performed the original ZCK authentication protocol can be used which ensures that this level of trust in the other's identity is maintained[3]. Thus an exchange of keys that can be used for the ZCK authentication protocol for future identification or message authentication processes is included in our new scheme. For each communication pair A and B the certificate exchange only needs to be performed once for proving identity whereas later on only the original efficient ZCK authentication needs to be executed for identification and message authentication based on the identity proven level of trust. In some sense the protocol can also provide non-repudiation as explained below.

[3]The ZCK authentication scheme improves or maintains the level of trust between two entities. Since the proof of identity is the highest trust level we can achieve, in this case the ZCK authentication maintains this level.

The idea of the protocol is to divide time into intervals and let a set of keys of a key-chain only be valid for one time interval, similar to the idea of TESLA [5]. Let $h(x_i) = x_{i+1}$ be a hash-chain, x_0 be the anchor of the chain and $x_{2(n-1)+1}$ be the final element of the key-chain. We let two keys of the chain be valid in a time interval. Let t_i be a point in time which we assume to be the starting time, L be the length of the time intervals, and I_{j-1} be the j-th time interval. We assume that time is divided into time intervals of length L, i.e., there are time values t_i, t_{i+1}, t_{i+2}, and so on, such that the time difference between t_j and t_{j+1} is L, and such that the time difference between t_j and t_{j+m} is mL. The k-th time interval I_{k-1} lasts from t_{i+k-1} to t_{i+k}. Let d_s be the time difference measured in interval lengths L between the current time t_c and the time t_s, i.e., $d_s = c - s$. Assume the starting point is t_i and the keys valid during the corresponding interval I_0 are $x_{2(n-1)}$ and $x_{2(n-1)+1}$, i.e., there are keys for n time intervals in the key-chain. This fact is depicted in Figure 1. Then $x_{2(n-1)}$ and $x_{2(n-1)+1}$ are keys for the first time interval I_0 between time t_i and t_{i+1}, x_{2j} and x_{2j+1} are keys valid for the time interval I_{n-1-j}, and x_0 and x_1 are the keys valid in the final time interval I_{n-1}. Furthermore the interval difference between the current time t_c and starting time t_i is $d_i = c-i$ such that the current time interval is I_{d_i}, and the keys valid for this interval are $k_c = x_{2(n-1-d_i)}$ as well as $k_{c+1} = x_{2(n-1-d_i)+1}$.

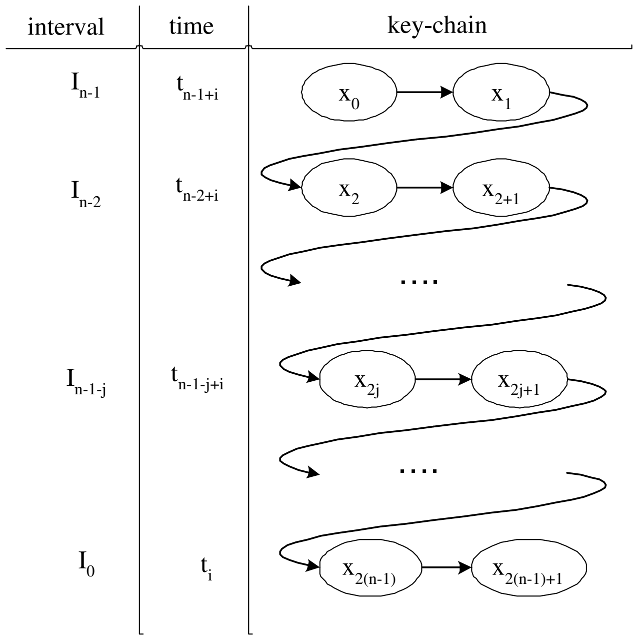

Figure 1: Time intervals for key-chains.

Assume Alice wants to identify and authenticate to Bob. Then Alice holds a secret anchor x_0^A and the public value $PK^A = x^A_{2(n_A-1)+1}$. Bob likewise holds x_0^B and $PK^B = x^B_{2(n_B-1)+1}$. The public values are signed by the CA at some time interval t_u and t_v, respectively, such that Alice and Bob obtain certificates $< t_u, PK^A >_{CA}$ and $< t_v, PK^B >_{CA}$.

Now, if Alice wants to prove to Bob her identity she sends Bob her certificate. Assume Bob determines the current time as t_j^B and Alice as t_i^A. These two timings might be different since we do not require a strict time synchronization. Then Alice and Bob determine the time difference between the current time they measure and the certificate time such that they obtain the interval differences $d_u^A = i - u$, $d_v^A = i - v$, $d_v^B = j - v$, and $d_u^B = j - u$. Alice then opens the key $k_{c+1}^A = x^A_{2(n_A-1-d_u^A)+1}$. Bob now checks whether $h^{2d_u^B \pm \{0,1\}}(k_{c+1}^A) = PK^A$ where $h^i(x)$ is the i-th iteration of the hash function. We allow that Bob's determined time t_j^B differs from Alice's determined time t_i^A by one interval length L, i.e., Alice's and Bob's determined times may each differ from the actual global time by L.[4] Hence Bob will tolerate that he got a chain element that is one position behind or before the element he expected. We express this as a correction term '$\pm\{0,1\}$' in the formula. To make the protocol secure we use further techniques similar to the original ZCK authentication key-chain scheme.

We now want to reduce the risk that Bob uses key values he just obtained to impersonate Alice (or vice-versa). To prevent such an attack where Bob obtains the key-chain value for some time interval, and then uses it to a third entity we introduce a random seed. Alice and Bob select a random seed r_A and r_B. Both Alice and Bob have a set of pre-computed certificates in the form $< t_A, r, PK^A >_{CA}$ and $< t_B, r, PK^B >_{CA}$ for all possible r which they obtain from the CA while being connected to a server. Now Bob can only impersonate Alice to a third party John if John uses the same random seed as Alice. Since this is an interactive proof system we expect that a small set for the seeds r suffices, e.g., r being an 8-bit number. In that case each entity had to store 2^8 pre-computed certificates. If we set the time intervals to be 1/2 minute[5], Bob had to ask 2^7 entities on average in one minute to find an entity using the same random seed as Alice. We can easily prevent that Bob asks the same entity multiple times in a short time-interval by introducing an exponentially growing pause interval after an unsuccessful authentication process. Further possibilities to increase the security level are given below. Assume Alice wants to authenticate to Bob and starts the authentication process. The protocol then works as follows:

[4]Note that the required level of time synchronization only depends on the size of the time intervals.

[5]If time is synchronized every time a server connection is established this is a very generous value which might be reduced to increase the security level.

```
 0 : $A$ sends a $hello$ message to $B$
 1 : $B$ sends a random seed $r_B$ to $A$
 2 : $A$ chooses a seed $r_A$ and sends $r_A$ as well
     as the pre-computed certificate
     $< t_u, id_A, r_B, PK^A >_{CA}$ with $PK^A = x^A_{2(n_A-1)+1}$
     to $B$
 3 : $B$ verifies the signature of the CA, and
     then sends the pre-computed certificate
     $< t_v, id_B, r_A, PK^B >_{CA}$ with $PK^B = x^B_{2(n_B-1)+1}$
     to $A$
 4 : $A$ verifies the signature of the CA
 5 : Assume $A$ determines current time $t_i^A$ with
     $d_u^A = i - u$ and $d_v^A = i - v$, and $B$ determines
     current time $t_j^B$ with $d_v^B = j - v$ and $d_u^B = j - u$
 6 : $A$ chooses a key-chain with final element
     $y^A_{n'_A}$ and sends authenticated session key for
     ZCK authentication protocol with time key
     $k_c^A = x^A_{2(n_A-1-d_u^A)}$ as $(y^A_{n'_A})_{k_c^A}$ to $B$
 7 : $B$ chooses a key-chain with final element
     $y^B_{n'_B}$ and sends authenticated session key for
     ZCK authentication protocol with time key
     $k_{c+1}^B = x^B_{2(n_B-1-d_v^B)+1}$ as $(y^B_{n'_B})_{k_{c+1}^B}$ to $A$
 8 : $A$ opens her key by sending $k_{c+1}^A$ to $B$.
 9 : $B$ checks if $h^{2d_u^B \pm \{0,1\}}(k_{c+1}^A) = PK^A$
10 : $B$ opens his key by sending $k_{c+1}^B$ to $A$
11 : $A$ checks if $h^{2d_v^A \pm \{0,1\}}(k_{c+1}^B) = PK^B$
12 : $A$ opens her key by sending $k_c^A$ to $B$
13 : $B$ checks if $h(k_c^A) = k_{c+1}^A$
14 : If any check fails, $A$ and $B$ stop execution.
15 : $A$ and $B$ store the certified session keys
     $y^B_{n'_B}$ and $y^A_{n'_A}$
16 : $A$ and $B$ use the certified session keys for
     message authentication as described in the
     ZCK authentication protocol.
```

Remarks:

- The scheme 'inherits' the requirements of the ZCK authentication scheme. Especially, it has to be ensured that the messages (the session keys) in Steps 6 and 7 were received. This can easily be done by a serial processing of all the steps.
- Once a session key of a key-chain is exchanged and authenticated this key is used as described in the basic ZCK authentication scheme, i.e., it is used for all further identification and message authentication procedures for the entity pair Alice and Bob.
- There might be more messages authenticated in an identity certified manner than just the session keys. In that case Steps 6 and 7 are enhanced such that they consist of all messages that are to be authenticated. All messages are then authenticated by the same key as it is used in Steps 6 and 7.
- The scheme does not allow non-repudiation by itself. However, in case that a trusted time-stamp service is available non-repudiation can be introduced by send-

ing messages immediately to the time-stamp service. However, since this requires a connection to a server and is quite expensive it might only be done in special cases, e.g., for important messages or to prove that an entity acts malicious.

- For the public-key certificate an arbitrary signature scheme might be used. We assume an RSA like scheme that performs very efficient in the verification step. However, further improvements might be possible by using a signature scheme that performs verification even faster than RSA.
- It is possible to reuse a certificate at different times. Hence it is sufficient to have one certificate for each random seed r per device. The probability that the same certificate is used to prove identity to two different entities at the same time is very small.
- It is possible to replace the certificates by values that were signed by the user itself. This saves the connection to the CA but still requires a temporary connection to a workstation to perform the signature generation. However, it slows down the protocol since a certificate that assigns a public key to the user has to be sent first.
- For a higher security level it is possible to perform several authentication processes in parallel. In such a case all messages of the parallel processes are sent at once such that the number of exchanged messages does not increase whereas the data overhead and computational overhead increases linearly.

Although TESLA [5] as well as our approach couple Lamport's hash-chains with a loose time synchronization mechanism there are major differences. TESLA provides authentication for broadcast streams. One single key-chain is generated at the sender side, and in each time interval elements of the key-chain are revealed to the receiving parties. Such revealed key-chain elements are applicable to data sent in a particular time interval some time before, namely the key disclosure delay. Implicitly this means that TESLA uses one valid chain element for each time interval. All data which is transmitted within the same time interval is authenticated by the same element of the chain as temporary valid message authentication key. Contrary, our scheme aims at mutual agreement on session keys for unicast traffic. More precisely, the scheme initially exchanges the final elements of key-chains between two parties to subsequently start the ZCK protocol for re-recognition of the involved parties. Two key-chains, one generated by Alice and the other one by Bob, are applied. Our scheme links knowledge of the key-chain's anchors to the entity's identity, and furthermore provides a key-exchange for the subsequent ZCK protocol. This subsequent ZCK protocol phase is independent of the time intervals.

3.1 Operational Complexity

We consider the complexity of above scheme for the identity certified exchange of session keys. We do not consider any further message authentication here that might be done by the ZCK authentication scheme. We take a look at the computational overhead, the data overhead, data storage, and necessary number of messages.

1. *computational overhead*: each entity performs one signature verification and several applications of a hash-function which we consider to be negligible in cost compared to a public-key operation as argued before.

2. *data overhead*: the overhead that is induced into the network traffic for a mutual authentication are two seed values, two certificates, and seven hash values. Let s be the seed size, l a signature size (without the clear-text message), c the certificate size, and w the hash size. Then the data overhead evaluates to $2s + 2c + 7w$ bytes. Assuming 4 bytes each for a time value t and an id value, we obtain a certificate size of $c = 8 + s + w + l$ bytes. Hence overall the data overhead is $4s + 2l + 9w + 16$ bytes. For scenarios that we envision, we assume $s = 1$, $l = 128$, and $w = 10$ bytes. Hence the overall overhead evaluates to 366 bytes.

3. *data storage*: to store all pre-computed certificates there is storage of $2^{8s} \cdot c$ needed. For above parameters there is space of $2^8 \cdot 147 = 37632$ bytes needed.

4. *number of exchanged messages*: without the initial Step 0 there are 8 packets sent in a serial fashion.

On platforms where there is not sufficient memory storage for 2^8 pre-computed signatures available it is possible to lower the seed size and perform several parallel identification steps. For instance, by choosing a seed size of 4 bits and 2 parallel processes, the security level stays the same whereas the data storage is reduced to $2^4 \cdot 147 = 2352$ bytes. Each entity now has to perform 2 signature verification steps, and the data overhead doubles to 724 bytes whereas the number of exchanged messages stays the same.

3.2 Security

We now prove the security of our scheme. We start by the following corollary which states that our scheme is secure against an active adversary in an ideal case. First note that above protocol is the same as the ZCK authentication protocol where we force that an authentication process for a given key-chain (as determined by the certificate) is either successful, or another key-chain will be used. Hence we avoid the loop (Step 8 in Section 2.1) in the original ZCK authentication protocol. Let l_r be the security parameter of the random seed r (its bit-length), and l_h be the security parameter of the message authentication code as well as the hash function (the bit-size strings are mapped to). By assuming that there is perfect time synchronization such that the interval length can be set infinitely small, and that $l_r \geq l_h$ it immediately follows that our scheme is secure against active adversaries. It is also clear that it is secure against a man-in-the-middle attack.

COROLLARY 1. *Assuming that the used signature scheme of the CA is unconditionally secure, that there is perfect time synchronization such that the interval length is infinitely small, and that $l_r \geq l_h$, our scheme is as sound as the non-forgeability of the underlying message authentication function against active adversaries.*

PROOF. *As proven in [6], the ZCK message authentication protocol is secure against active adversaries. Above protocol uses the message authentication property to exchange the final element of a key-chain (Steps 6 and 7). By the assumptions it immediately follows that above scheme is as sound as the ZCK message authentication scheme.* □

It is clear that loose time synchronization as well as a small security parameter l_r for r introduces weaknesses. We see the following two attacks. An adversary Mallory starts an authentication process with Alice to obtain her certificate and a key of the key-chain for the seed r_A. He then uses these values for Bob to obtain his certificate and one of Bob's keys for the seed r_B. Now he uses Bob's key and certificate against Alice who has to use the same seed r_A again such that the adversary obtains the second key for the current time interval. Mallory is then able to impersonate Alice against Bob in the current time interval. In the second attack, Mallory eavesdrops an authentication process between Alice and Bob to obtain a certificate and the keys for the current time interval. He then tries to impersonate Alice against a third entity John with these values in that same time interval. In both cases Mallory uses the fact that he knows keys that are valid in the current time interval. Hence he has at most two time intervals to impersonate Alice because of the time tolerance we allow. However, in both cases Mallory has to find an entity that uses the same random seed r_A as Alice since the keys he obtained are only valid for this seed. It becomes clear that by increasing the bit-size of the seed r and shortening the time-intervals the security level can be set arbitrarily high. In practice we believe that a time interval of 1/2 minute and a seed r of 8-bits suffices. In that case Mallory had less than 1 minute time left to start an authentication process with $2^7 = 128$ entities on average. Especially if we assume that devices are put every night in their cradle to charge the battery it is appropriate to assume much smaller time intervals which decreases the possibility for a successful attack. Although it is hard to determine an exact level of security for our scheme we believe that the parameters can be chosen in a way that the probability of success for an adversary is negligible. However, it should be clear that our scheme does not have a high security level in a formal way. Hence it is not appropriate for applications such as financial transactions of significant value. As we outlined before we see its application for routing protocols and peer-to-peer applications.

For a higher security level it is possible to perform several authentication processes in parallel. For instance, when performing three processes in parallel, an adversary Mallory had one minute given to find an entity requiring the proper nonce which on average takes him $2^{23} = 8388608$ authentication processes. For an interactive proof system this can be considered to be a sufficient level of security even for sensitive areas. As said before, by introducing an exponentially growing time pause after a protocol failure it is easy to prevent that an adversary starts several authentication processes with the same device.

4. RATING OF THE SCHEME

As noted before, it is also possible to perform identification with a public-key scheme as well as a symmetric-key scheme. In this section we compare our new scheme to these schemes. First we briefly describe a public-key scheme and a symmetric-key scheme. Both the schemes provide mutual identification.

In a public-key scenario each entity has a certificate *cert* issued by a CA and an assigned public/private key pair. An authentication process works as follows.

```
1 : $B$ sends a random number $r_B$ to $A$
2 : $A$ signs the random number and sends
    $r_A, (r_B)_A, cert(A)_{CA}$ to $B$
3 : $B$ signs $A$'s challenge and sends
    $(r_A)_B, cert(B)_{CA}$ to $A$
```

In a symmetric-key scheme there is a trusted server S which establishes the trust relationship between A and B. Each entity shares a secret key with the server. First, A asks S to establish a relationship to B. Then S sends A and B a session key K which these can use for authentication. We denote the encryption of a message m by a key S as $E_S(m)$.

```
0 : $A$ sends a hello message to $S$
1 : $S$ sends $E_A(K)$ to $A$
2 : $S$ sends $E_B(K)$ to $B$
3 : $A$ sends $r_A$ to $B$
4 : $B$ computes the MAC over $r_A$ and sends
    $(r_A)_K, r_B$ to $A$
5 : $A$ computes a MAC over $r_B$ and sends $(r_B)_K$
    to $B$
```

For the comparison we consider the identification of an entity. This is reflected by the three protocols as shown above. Table 1 gives an overview over all three instantiations. The first rows describe the complexity whereas the following rows describe features. Note that the values for the public-key scheme depend on the used scheme. Where applicable, a 'x' means that the scheme provides a feature, a '-' means that it does not, and a 'o' means that the scheme can provide the feature with modifications (at higher computational cost).

For the public-key and symmetric-key scheme we assume that all nonces are 10 bytes in size. The values for the key-chain scheme are obtained as in Section 3.1. For the public-key scheme we assume that a signature (without the

Table 1: Overview of mutual authentication schemes.

	Key-Chain	Public-Key	Sym.
public-key size (bytes)	37632	264	10†
exchanged messages	8	3	6
exchanged bytes	366	548	60
computational effort	2 Ver.	2 Sig., 4 Ver.	0
authentication	x	x	x
non-repudiation	o	x	o
key-exchange	-	x	x
signature	-	x	-

† size of the shared key

signed message) is 128 bytes long (such as 1024-bit RSA) and that a certificate is 136 bytes in size (4 bytes each for time and ID, and 128 bytes for the signature). There are two certificates and two signatures exchanged, as well as two nonces. Overall, there are two signatures generated and four signatures verified (including the certificate verification). The symmetric-key scheme is the most efficient one but it requires a permanent server connection which might be suitable for military scenarios or for application in a private home. The public-key scheme requires computationally demanding operations that are significantly more expensive than all other schemes but provides non-repudiation as well as a key-exchange.

Our new scheme is nearly as efficient as the symmetric one without its shortcomings. It is not capable of providing a signature or a key-exchange, but comes with a mutual identification at a low complexity. It is extremely efficient in providing identification in the view of computational as well as network resources at the cost of some data storage. Furthermore, it is compatible with our ZCK message authentication scheme which can then be used at very low cost in further identification and message authentication processes based on the proof of identity by the IC scheme. Hence it is suited as a building block for protocols in ad-hoc networks consisting of devices with moderate computing power such as cell phones, PDAs, and embedded devices.

5. CONCLUSIONS

In this work we presented a mutual identification scheme which mainly requires symmetric primitives such as key-chains and avoids exhaustive computations. The scheme is capable of providing the same functionality as a MAC scheme that uses a symmetric key as determined by a key-exchange protocol which also ensures the proper identification of the entities. Our scheme can be used to exchange a key-chain element which then in turn is used for further extremely efficient identification and message authentication processes. However, our scheme requires a loose time synchronization as well as a temporary server connection and some data storage for pre-computed values. Its security can only be proven in an ideal case and it is heuristic in practical applications.

We believe that our scheme shortens the gap to digital signature schemes that are still more flexible and powerful but also computationally demanding. We think that for ad-hoc and pervasive networks which mainly consist of computationally weak devices protocols have to be designed that are largely based on symmetric primitives while getting close to the functionality of public-key schemes. Only such protocols will allow that ad-hoc and pervasive networks are deployed with a sufficient level of security.

6. REFERENCES

[1] A. Bosselaers. Even faster hashing on the pentium. In *Rump session of Eurocrypt'97*, 1997.

[2] E. De Win, S. Mister, B. Preneel, and M. Wiener. On the performance of signature based on elliptic curves. In *Proceedings of ANTS III*. LNCS 1423, Springer-Verlag, 1998.

[3] Y.-C. Hu, A. Perrig, and D.B. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of the Tenth Annual Network and Distributed System Security Symposium (NDSS 2003)*, 2003.

[4] A. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[5] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. 2002.

[6] A. Weimerskirch and D. Westhoff. Zero common-knowledge authentication for pervasive networks. In *Proceedings of Selected Areas of Cryptography 2003 (SAC 2003)*, 2003.