A Strong Two-Way Authentication Method for Low Capacity Solutions

Gökhan DALKILIÇ, H. Şen ÇAKIR, M. Hilal ÖZCANHAN

Abstract— **One of the widely known challenge - response authentication methods is the two–way authentication. First two-way authentication methods were limited by those days' capacity of processing power and had limited security. The later protocols and standards on the other hand were too broad and resource demanding because they aimed at addressing all of the security demands of the "future"- as seen in those times. With the availability of new techniques and new tools at low costs, a finer granularity on the security of low capacity, low cost environments has become possible with good cost/performance ratios. Of course these solutions need careful design and functional definition, before being used in small designs. Our alternative two-way authentication is a step towards giving more meaning to the parameters suitable for low cost solutions, instead of using public-keys or certificates and still have a key exchange protocol as secure as others.**

Keywords — **authentication, mutual-authentication, low-cost devices, security.**

I. Introduction

To pay respect to Diffie Hellman let us remind ourselves that they were the fathers of two-way authentication. Even though the first Diffie Hellman authentication[1] was susceptible to a man-in-the-middle attack, some considerable time passed until this vulnerability was tackled and remedied for.

Up to present day, cryptographers are still working on two-way authentication to make it stronger against all known attacks. To this end today's popular public-key and public certificate tools are utilized. But these tools are known to cost in terms of ownership, processing, storing and protection. Relying on abundant resources in applications is nothing but seeking a forceful solution to counter-measure the vulnerabilities of two-way authentication. To give an example, designers are pushing the use of public certificates for authentication in small scale devices and even in wireless solutions. To exaggerate the abuse of resources, the certificates are required in every step until the whole transaction finishes.

Every network and hardware expert knows that passing certificates attached to each and every step of the transaction is a costly act in terms of network traffic and processing power. To cut the costs of the network load, some institutions using certification technologies opt to use their own local certificate authorities. This puts the whole operation in jeopardy, since hacking intermediary certificate authorities and public-keys are among today's new hobbies. In addition, the cost of premises and user protection of such certificate servers do not match the efforts spent in the design of the authentication tool. Designing a more appropriate authentication method and spending less on servers would be a much better solution.

One vulnerability of public certificate authentication is that people rarely question the validity of the certificate of their transaction partners. The awareness of questioning the validity of the public certificate of an applicant is reduced to a just "click ok" action by many users. Such care free applications cannot be accepted as secure.

We attempt to address these issues by providing a low cost alternative to the public-keys and certificates in two-way authentication, robust enough to make two-way authentication more cost effective in advanced systems and more affordable in low cost, low capacity solutions.

II. Present Solutions and Conformity to Standards

The flaws of Diffie Hellman key exchange; conversation in clear text and the full reliance on a stranger's identity have been tackled by many. Oakley key exchange[2] is one of the most famous of them all. However, it is commonly accepted that it is computationally very intensive and requires a lot of resources. Of course, there are other work adding extra security to the original Diffie Hellman key exchange, but there is a need for some standards to provide a common ground for the basic criteria that everybody should accomplish to claim that their two-way authentication method is secure. The standard in this area is the X.509 standard[3] which is discussed below.

A. The X.509 Standard

Standardization authorities trying to put some standards to one-way, two-way and three way authentication protocols have come up with a standard for two-way authentication, too. This is the X.509 Strong Two-Way Authentication standard and must be studied here, to prove that our proposed alternative is in conformity with the standards and does not open up new flaws by over simplifying some parameters or steps.

The X 509 Strong Two-Way Authentication standard – shown in Fig. 1 - asks the initiator A to use a time stamp, a nonce , the ID of the responder, sign all of these in a new entry and pass the proposed session key K_{ab} encrypted with the public key of the responder B. Similarly the responder has to send its own timestamp, nonce and ID. B also has to sign the nonce and its ID

Manuscript received November 10, 2008. Date of paper submission November 10, 2008.

Gökhan Dalkılıç is with the Computer Engineering Department, Dokuz Eylul University, Izmir, 35160 TR (e-mail: dalkilic@cs.deu.edu.tr).

H. Şen Çakır is with the Computer Engineering Department, Dokuz Eylul University, Izmir, 35160 TR (e-mail: sen@cs.deu.edu.tr).

M. Hilal Özcanhan is with the Computer Engineering Department, Dokuz Eylul University, Izmir, 35160 TR (e-mail: hozcanhan@cs.deu.edu.tr).

and send it to A, together with its own secret encrypted with A's public key. The standard is well balanced in terms of doing the same computations and exchanging similar parameters, by the partners. However, there are a few points worth pointing out here. The first point is although the ID of the responder is present, the ID of the initiator is missing in the challenge.

That piece of information is needed for two reasons: To be sure that the first message indeed arrived from the initiator and to protect the responder from DoS attacks[4]. The signature of values and the encryption-decryption processes involve intensive computations and assumes that both parties have the necessary capacity to do the calculations fast. Another point is the assumption that each side has the public-key of the other beforehand[4]. It is obvious that even if the two sides are total strangers to each other, there is a mediator -the certificate authority- that authenticates the two sides to each other. But the question arises "Do we really need certificates, signatures or public-keys in low cost-low capacity solutions to make an exchange secure?". We don't because the low cost solutions cannot make the computations of certificates easily.

A ➔ B : { t_A, r_A, ID_B, sgnData, E [PU_b, K_{ab}]}

B ➔ A : { t_B, r_B, ID_A, sgnData, E [PU_a, K_{ba}]}

Notation :

t_A , t_B	: timestamp of A and B, optionally expiration time also incl.
r_A , r_B	: nonce, pseudo-random number of A and B.
ID_A , ID_B	: Identity of A and B.
sgnData	: signature of A or B over parameters (t_X , r_X , ID_X)
PU_a , PU_b	: Public-key of A and B.
K_{ab} , K_{ba}	: Secret parameter of A sent to B and B to A.

Fig. 1. The X.509 Strong Two-Way Authentication standard. © Copyrighted, Cryptography and Network Security [4] , by William Stallings.

B. Can certificates, public- keys and signatures be replaced by other secure parameters and algorithms in low cost solutions?

We may rephrase the question as "Can low capacity solutions with shared secrets exchange secure session keys too, without the need for certificates?" The answer is "Yes, they can." With the advances in integrated devices and the drop in their prices, new technologies make it possible to have the secure secrets needed for two-way authentication. When used properly, these technological devices can match the security of the expensive certificates and public-keys, as we will show shortly.

A note of caution before indulging in details though: In the X.509 standard, the parameters are used to calculate the session key and provide additional security. However, the parameters can be used to calculate the *key* to encrypt *the session key* before exchanging it, if that "interim" key is secure than the authentication is as secure as the others.

III. THE PARAMETERS OF THE PROPOSED TWO-WAY AUTHENTICATION

First of all, the parameters of the proposed two-way key exchange will be outlined. Our proposed two-way authentication is shown in Fig. 2. Here also five parameters are passed, in the challenge. Let us remember that the motive is to add more meaning to the parameters. The timestamp, the nonce and the ID of the responder are preserved. However, the ID of the initiator will also be added for denying DoS attacks and for informing B about whose secret tables (which secrete to share) to use. Then, three of these four parameters will be hashed together with a secret shared by both parties and will be sent for B to compare the clear text with the hashed version. The computation of the secret $secret_{Ax}$ used in the challenge will be explained below.

---

1. A ➔ B {t_A, r_A, ID_A, ID_B, h(r_A, ID_A, ID_B, $secret_{Ax}$)}
2. B ➔ A {t_B, r_B, ID_B, ID_A, h(r_B, r_A, $secret_{Ax}$), E(K_Φ , K_s)}

Notation :

t_A , t_B	: timestamp of A and B, optionally expiration time also incl.
r_A , r_B	: nonce, pseudo-random number of A and B.
ID_A , ID_B	: Identity of A and B.
$secret_{Ax}$	: shared secret , comes from Table A
PU_a , PU_b	: Public-key of A and B.
K_Φ , K_s	: Calculated encryption key and secret parameter of B sent to A.

Fig. 2. The alternative Two-Way Authentication proposed.

In the response there are six parameters as opposed to the five of X.509. Again the timestamp, the nonce and the ID of the challenger are preserved. However, the ID of the responder will be added for preventing DoS attacks and for informing A about whose secret tables to use. Again the clear text parameters, excluding timestamp, will be hashed together with the same secret $secret_{Ax}$ and will be sent to A to compare the clear text with the hashed result. Then there is an additional parameter where the session key is passed in encrypted form. Let us now study the details of the exchanged parameters, because compared to X.509 standards some parameters have additional roles, some parameters are missing and there are unexplained shared secrets.

A. The ID Parameters: ID_A and ID_B

The ID parameters have always been the weak point of challenge-response authentications. Our suggestion, this value is strengthened for both by the inclusion of a unique serial number (S/N_X), by concatenating the IP address (IP_X) with S/N_X (1).

$$ID_A \equiv IP_A \,||\, S/N_A \quad ; \qquad ID_B \equiv IP_B \,||\, S/N_B \qquad (1)$$

Obviously, there will be an additional security in identities because they have become unique and dynamic. Thus, when a party changes its address, it can be traced and a peculiar IP address for a device may mean a masquerade attack and thus ring some alarms.

The Serial Numbers S/N_A and S/N_B are coming from a secure memory location[5,6]. Another property is to help the exchange partners match the serial number of its reciprocal to the secret tables A and B of the same reciprocal, which will be used during the key exchange. In other words, every device has a unique pre-lasered Table A and a re-writeable Table B. Fig. 3 shows the features of a commercial product[5] where the product has a unique, factory-lasered 64-bit serial number used as an input to a hashing engine. More information on that technology and tables is given in section V. Let us for now accept that we can have a unique and dynamic ID for our partners A and B. In Fig. 3 only a one-way authentication commercial product is depicted for simplicity but the same device can be used for two-way authentication; since B has what A contains, on a piece of paper

supplied by the manufacturer. The principle is still the same: A device with a unique serial number hinting at unique static and dynamic secrets, that it shares with B.

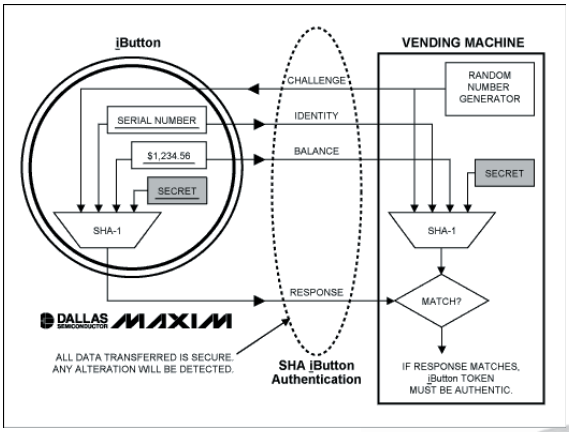

Fig. 3. A typical commercial application of a serial number in an authentication process. © Copyright Courtesy of Dallas Maxim Semiconductor [5].

B. The Timestamps and Nonces t_A, t_B and r_A, r_B

These parameters are the same as those used in the X.509 standard. There are no alternative or additional properties of timestamps. The use of the nonces is the same but there is an additional role for each of the nonce values. These additional roles and the regular usage will be explained in the section where the mechanism of the key exchange is outlined.

C. Instead of Signatures, The Secret(s) of A

Parameters utilized in the X.509 standard are agreed upon beforehand, that is parties share some information before they start the exchange. They know that the last parameter passed is encrypted with a public-key. The most important issue is the strength of the public-key encryption. We claim that if the encryption key K_Φ used in our design is as secure as the public-key of the X.509 standard, than our alternative authentication is also as secure. Furthermore, in X.509 there is only one private key, in ours there are many secrets leading to many keys.

In Fig. 4 there is another commercial product[7,8] where an internal 8 byte secret is input together with other parameters into a hashing engine, namely a SHA-1engine[5,7,9,10,11]. This example shows that it is possible to have a secret inherent in a low cost system, which can be used together with the other secrets as an input to a hashing function. The hash function is sometimes accepted as a raw version of a signature. The important issue here is to make sure that the secrets stored in the device can be trusted. This is tackled later in section V. Secret protection requires that the internal secret never leaves the system. The best security is to keep the secret inside the processor while using it to hash the inputs. For our purposes we assume that today's manufacturers guarantee to provide us with many secrets of satisfactory lengths inside a low capacity device; i.e. eight different 64 bit secrets. Therefore, we can imagine having a table of eight different unique static secrets. Let this be the Table A of Fig. 5.

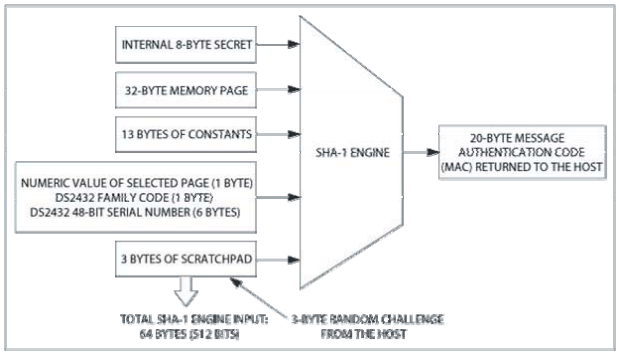

Fig. 4. A typical commercial application of a secret number as an input to a hashing engine, in an authentication process. © Copyright Courtesy of Dallas Maxim Semiconductor [7].

Table B will be unrevealed shortly.

pointerA#	Table A	pointerB#	Table B
1	$secret_{A0}$	1	$secret_{B0}$
2	$secret_{A1}$	2	$secret_{B1}$
3	$secret_{A2}$	3	$secret_{B2}$
4	$secret_{A3}$	4	$secret_{B3}$
5	$secret_{A4}$	5	$secret_{B4}$
6	$secret_{A5}$	6	$secret_{B5}$
7	$secret_{A6}$	7	$secret_{B6}$
8	$secret_{A7}$	8	$secret_{B7}$

Fig. 5. The static and dynamic secret tables of A.

Of course, the manufacturer supplies the unique Table A during delivery; thus only the responder should have a prior copy of the secrets. In our alternative we will not just use Table A's secrets but other inputs as well to make a hash, e.g. an entry from Table B. That mechanism and Table B is explained below, in detail.

The hash of the two parties' IDs together with the first secret entry of Table A looks very similar to the signed data of the X.509 standard. But our alternative is cheaper in every respect.

D. Instead of Public Keys: Table A & Table B of party A.

Looking back at our proposed key exchange shown in Fig. 2 the key K_Φ that is used to encrypt the session key is in the heart of our design. While in other key exchanges the session key is co-generated, in our design the session key is generated by the responder B. The key that encrypts the session key; K_Φ, is co-generated instead. Let us see the inputs to K_Φ. In the previous section, we had introduced a shared secret from a table while calculating the hash of the challenge parameters. So we already have Table A of Fig. 5 with eight secrets, written once and only read internally. At the very beginning of the key exchange, when the two parties have just initialized, A starts using the first entry in Table A; that is $secret_{A0}$ to calculate the hash in the challenge. Moreover, A points to a secret from Table B using its nonce r_A in the challenge. B uses the secret $secret_{Bx}$ pointed by A as one of the inputs to calculate K_Φ. How B calculates the value of x in $secret_{Bx}$

is left to the explanation of the exchange mechanism in section IV.

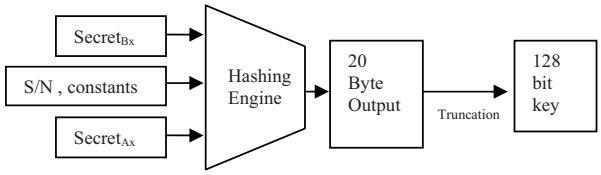

Fig. 6. Calculation of key K_Φ.

Some commercial eeproms and secure microcontrollers also supply a memory page that can be filled by the user. Thus, the user can create more secrets in another protected table, over just one wire (called 1-wire [9,10]). This technology is explained briefly in section IV. In our case, let us assume that Table B of Fig. 5 is the table we created. B uses this shared secret, e.g. $secret_{B6}$, some constants, $secret_{A0}$ as input to the same SHA-1 function and truncates the 20 byte outcome to desired key length of 128 bits, as shown in Fig.6. Since a SHA-1 engine is already used to hash the challenge parameters, why not use the same hashing engine SHA-1for calculating K_Φ. Obtaining K_Φ by using manufacturer made static and user defined secrets, remove any doubts on the static secrets. Since K_Φ is calculated by using the eight by eight secret tables many would think that the value of K_Φ is limited to a possibility of 64. That is not true. The outcome of the SHA-1 engine is 20 bytes and K_Φ is obtained by truncating this value, which makes K_Φ more random. By making the truncation method complicated a huge number of K_Φ values can be obtained. One simple way may be, looking at the last digit of the nonce r_A. E.g. if it is 0: take the right-most 128 bits of the SHA-1 outcome; if it is 1: take the leftmost 128 and if it is 2: take the mid 128bits. Even this simple trick yields a 3x128 possible values of K_Φ.

Also observe that K_Φ is never passed on the communication line, a property of the session key K_s in the other key exchanges. Therefore there is no loss of property in our suggested alternative. The explanation of how the entries from Tables A and B are chosen when the old one expires is left to section IV below, when the mechanism is elaborated on.

E. The session key

The key K_s that is the heart of other designs is a relaxed issue in our case. It is simply generated keeping in mind all the recommendations given by cryptographers. Its length, uniqueness and other properties are left to the choice of the solution owner. But since there is a SHA-1 engine in our example[5,7]; maybe a SHA-2 in a more expensive solution, this same hash function can be used to calculate K_s. Any preferred entries from tables A and B, together with the serial number, a random number (seed) and any other entries in the rest of the secure memory can be the input to the hash engine. The 20 byte output can be truncated in any preferred way to obtain a 128 bit key K_s. The vital point is that this new session key K_s is encrypted using K_Φ before being sent to A. Commercial devices with an embedded AES encryption engine and a SHA-1 hashing engine are common today. Therefore, K_s can be encrypted using K_Φ with an AES engine and sent to A in the response.

IV. THE EXCHANGE MECHANISM

Our proposed two-way authentication alternative (Fig. 2) does not look much different than the X.509 standard which means we have not deviated far from the standard. However, some parameters have additional meaning and uses.

A. The Challenge

Let us start by studying the challenge. The first parameter, the time stamp t_A is the same as that in the X.509 standard.

A ➔ B $\{t_A, r_A, ID_A, ID_B, h(r_A, ID_A, ID_B, secret_{Ax})\}$

Fig. 7. The challenge of the proposed key exchange.

It consists of an optional generation and expiration time. Thus a delayed delivery of the first message can be detected and a retransmit request can be made by B.

The second parameter, the unique random number- nonce r_A- however has three roles instead of two. The commonly known role is to help detect replay attacks. B must compare nonce r_A with the previous nonces and reject any challenges with the same nonce. Thus nonces are not discarded but stored for later comparisons. The second role of nonce r_A is a challenge to party B to show both, that it has received this fresh nonce r_A untampered and will send a response proving it has the shared secret, $secret_{Ax}$. This will be explained in the response section. Additionally, as a third role *nonce r_A is a pointer to the $secret_{Bx}$* , one of the parameters used to calculate K_Φ. Since the nonce is a large random number, the pointer to $secret_{Bx}$ is calculated as follows (2), for an eight entry table:

$$x = (\text{lastdigit of nonce } r_A) \; mod \; 8 \; ; \text{therefore } 0 \le x \le 7 \qquad (2)$$

Assuming that the last digit of nonce r_A is 3, in the challenge, party A is telling party B to use the secret at location 3 of Table B in Fig. 6– which happens to be $secret_{B2}$. B uses this value to calculate K_Φ and uses K_Φ it to encrypt K_s as explained in the previous section.

The initiator A, also sends its identity ID_A – authenticated with its serial number, as an additional parameter compared to the X.509 standard. This parameter is absent in X.509 because A signs some parameters with its private key. B can only open it by using A's public key; hence thinking it must have definitely come from A. But this requires B to know the public key of A and make a computation to strip the signed information out of the signature and compare with that sent in clear text. We believe that this can clog low capacity devices, in case of a DoS attack. Our suggestion is to send the initiators identity ID_A in clear text, for B to be able to drop the call quickly, in case of an unexpected challenger.

The authentication of A's identity and the integrity check of the clear data are done in the next parameter. Identities are hashed together with the nonce r_A and $secret_{A0}$. And this is very similar to the signing process, in the X.509 standard. The hash has the $secret_{A0}$ as an input, which is never passed to B. Including A's nonce in the hash makes sure that if r_A was tampered on the way it will be detected by B, because the result of the hash will not match the clear text values that A has sent.

Finally, ID_B is necessary for B to check quickly if itself is the

intended address. If not, it simply drops the call.

The last point to mention is the missing parameter of X.509, in our alternative. In standard X.509, party A both starts off an exchange and at the same time forces a session key K_s in one go. It was mentioned before too that using public-keys as a means for exchanging secrets overwhelms low cost, low capacity solutions. But more critical than that; accepting a secret and the identity of a caller in the same step can be a flaw. Because again a malicious attacker can clog B. The final session key should be settled only in the second step, when parties really prove one to the other. After that parties can decide to share other secrets or more keys thereafter.

B. The Response

The responder namely B, upon receiving the call from A, looks at the timestamp t_A. If the message is old, it drops the call immediately and hopes that if it were a real call the sender syncs its time and makes a new call; after a random amount of time. Similarly, a check is run for the nonce r_A. If it is the same as the last message's nonce, again the call is dropped.

Next B checks the identity of the caller. B must have an idea about the IP address and the serial number of A. Assume that B has a table of serial numbers. B now identifies which secret tables A and B are to be used. If A is an expected caller B goes on to the next check which is the destination address. If B understands that the challenge is for itself, it goes to the final step of checking; otherwise again the call is dropped.

In the final step of checking, if it is the first contact with A after an initialization (more on this in section VI) B knows that A is using $secret_{A0}$ and calculates the hash which has four inputs. If the result of this hash is a match to the clear text data then A is indeed the party making the challenge and the challenge is new. Furthermore it is guaranteed that the contents of the challenge have not been tampered with, on the way.

B ➔ A $\{t_B, r_B, ID_B, ID_A, h(r_B, r_A, secret_{Ax}), E(K_\Phi , K_s)\}$

Fig. 8. The response of the proposed key exchange.

After making these checks B responds by sending its own timestamp for party A (Fig. 8) to decide if the response is old or not. Expiration time was sent by A and can be verified by resending it. Time synchronization will not be taken up here, but it is expected that if too many calls are dropped, the parties are expected to do a time sync.

The next parameter *nonce* r_B however also *has three roles* in our alternative. The first one is a check for A to see if the response is a replay or not. The second role is its presence in the hash for the parameters to be passed, to authenticate their integrity. Here B calculates the hash of its nonce r_B, A's nonce r_A and the secret $secret_{A0}$. B sends this hash value to A to prove that indeed it has received the fresh nonce of A, finished all checking and has the secret $secret_{A0}$ apriori, as expected. This authenticates that the responder is indeed B and gives A the chance to test if nonce r_B has been tampered with.

Thirdly, *nonce* r_B is a pointer to the next $secret_{Ax}$ for A, to use in the next key exchange challenge. Since nonce r_B is a large random number, the pointer is calculated as follows (3):

$$x == (\text{lastdigit of nonce } r_B) \bmod 8 \text{ ; therefore } 0 \le x \le 7 \qquad (3)$$

Assuming the last digit of nonce r_B is 6, in the response, B is telling party A that when the time expires and A has to make a new challenge, the secret at location 6 of Table A in Fig. 6 has to be used– which happens to be $secret_{A5}$. Thus, nonce r_B is a forecast to A about the next secret to be used in the next challenge.

The parameters ID_B and ID_A were explained in detail before. The only difference is their order, in the response. This is only a matter of syntax.

Finally, B computes K_Φ as described in section III using the secret $secret_{Bx}$ dictated by A's nonce r_A (as explained in the above section) and generates a session key K_s. B then encrypts K_s using a pre-decided algorithm, i.e. AES , and sends K_s to A, in an encrypted form.

A has both secrets $secret_{Ax}$ and $secret_{Bx}$. $secret_{Ax}$ dictated by B in B's nonce r_B, and $secret_{Bx}$ dictated by A itself. Thus calculates K_Φ , as well. After calculating K_Φ, A decrypts the encrypted message and recovers the session key K_s. That ends the two-way authentication process and the exchange of a secret session key. Observe that no numbers about K_Φ is sent to each other, but is obtained by either party, from their tables and through some calculation.

V. TECHNOLOGY OF OUR SECURE SECRETS

In this section the reliability of random number generators, static and dynamic secrets, hashing and encrypting engines supplied by manufacturers will be discussed. Although it is not the scope of this paper to detail on these technologies an effort will be made to persuade the reader that our reliance on these technological capabilities is sound.

The first concern is how secure the secrets recorded on the devices are, at the time of manufacturing. Manufacturers claim that they can prove their products contain encrypted information that only the issuing authority could have created[5,7], plus once written these secrets cannot be read from outside the device but only used as an input to the internal hashing engines. To fend off side-channel attacks, advanced tamper detection, special NV SRAM memories and electromagnetic radiation control technologies are used [7].

Another issue is the 1-wire property of commercial devices [5,7,9,10]. To reach the encrypted code space of a product only one wire is used and the whole of the circuitry need not be powered on. Manufacturers submit proof that tampering detection, encrypted memory and MAC authentication before read and write operations are well up to the security standards[7].

The third concern is whether the secrets are unique and only supplied to the buyer. This can be compared to the trust toward certificate authorities. Major manufacturers of security technologies build their industry on the trust they supply to their markets, therefore they are the secret authorities of their field.

Another issue is the quality of the hashing/encryption engines inside the devices[5,7,8,9]. Manufacturers claim that they guarantee both the uniqueness of the random numbers generated –an essential aspect of authentication, and the conformity of hashing function and encryption algorithm of their products to

standards [5,8,9]. In short, all inputs and engines involved in an authentication process can be on embedded on a secure device.

The technologies using public-keys and certificates have costs incurring from annual fees, processing power and fast connection to third parties. Our alternative, on the other hand has no such costs.

VI. HOW SECURE IS OUR ALTERNATIVE

The proposed alternative has to be strong against known attacks. In terms of a key exchange authentication our alternative doesn't have a security flaw. It is as strong as the hardware is. Therefore, the only vulnerability may lie in hardware side channel attacks[6]. To address this type of attack, some information about the efforts of the manufactures were given in section V, but that is not all. More information about the definition of side channel attacks and suggested remedies are given in the references[6], the evaluation of these side channel attacks will not be discussed any further here because they are the topic of a different argument.

The originality and thus the security of our alternative lies not in the generation of the session key to be exchanged, but in the composition of a key that will be used to encrypt the session key. No matter how strong our key might be the strength of our alternative depends on the encryption algorithm used to encrypt the session key. Once that algorithm is compromised or gets old, a new algorithm has to be employed in the key exchange process[8]. That is to say; we recommend to use AES instead of DES or 3DES for the encryption algorithm but it is quite possible that some day AES will be declared weak. So our authentication is as strong as AES and no more. The same argument is true for the SHA-1 hashing engine used. New products are coming out with SHA-512 engines, but for the time being they are not low cost.

As for DoS attacks, if the attacker intercepts the identity of A in the challenge and resends the challenge to B, he will not be able to clog B. Because, B uses the nonce or the IP address of A to drop the challenge. However, if the attacker spoofs the IP of A, as well as tampering with the timestamp and the nonce B discovers the attack after examining the hash value in the challenge. Just one hash calculation is not expected to clog B.

VII. FUTURE WORK

There are a few things that need to be done in the future. First of all instead of eight member secret tables, 64 or more secrets should become available[8]. Furthermore, the secrets can be 128 bit instead of 64 bits. But these are related with the chip manufacturers. Our future work may be related to making better use of the timestamp parameters, i.e. making them intelligent like we did for the nonces. It is possible to add more security by studying the timestamps as natural numbers as well.

One more future work is about the initialization of the two parties. It may appear as a problem if one of the parties initializes during or before the key exchange starts. How will one know that the other has initialized. This needs a full consideration because it can turn into a security flaw. But with some work a special timestamp or a new special parameter hinting toward an initialization can be utilized. Even the first key exchange after any initialization state can be discarded and the key at the end of the second key exchange process can be accepted as the valid key.

VIII. CONCLUSION

The offered method is for two-way authentication in limited capacity devices, in order to replace public-keys or certificates. As two parties know each other a key K_Φ can be shared. To increase the security of the key, the key has a lifetime. Each party has the same two distinct set of secrets. One is unique and inherent in the hardware and the other set is co-defined by both parties. Generated session key is encrypted by using K_Φ which is the output of the SHA-1 function.

The inputs of the SHA-1 which finally yields to K_Φ are the serial numbers, the nonces, addresses and the secrets from both sides. K_Φ is never transferred between the two sides and it makes the algorithm as secure as public-key cryptosystems.

REFERENCES

[1] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. on Information Theory, Vol. IT-22, November 1976, pp. 644-654.
[2] H. Orman, "The OAKLEY Key Determination Protocol", RFC2412, November 1998.
[3] C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure: Certificate Management Protocols", RFC 2510, March 1999.
[4] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone "Handbook of Applied Cryptography", CRC Press, 1996, Ch.12 pp511-512.
[5] Dallas Semiconductor of Maxim Integrated Products Inc., Application Note 1770, Securing Electronic Transactions Using SHA-1 Secure Hash Algorithm
[6] Dallas Semiconductor of Maxim Integrated Products Inc., Application Note 3824, "Security in Embedded Systems".
[7] Dallas Semiconductor of Maxim Integrated Products Inc., Application Note 3675, "Protecting the R&D Investment—Two-Way Authentication and Secure Soft-Feature Settings".
[8] Dallas Semiconductor of Maxim Integrated Products Inc., Application Note Application Note 152, "SHA iButton Secrets and Challenges".
[9] Dallas Semiconductor of Maxim Integrated Products Inc., Application Note Application Note 190, "Challenge and Response with 1-Wire SHA devices".
[10] Texas Instruments, SLUS696A–June 2006–Revised February 2007, "SHA-1/HMAC Based Security and Authentication IC with SDQ Interface".
[11] Texas Instruments, SLUA389A–July 2006–Revised October 2006, "How to Implement SHA-1/HMAC Authentication for bq26100".