How To Re-initialize a Hash Chain

Vipul Goyal
Department of Computer Science & Engineering
Institute of Technology
Banaras Hindu University
Varanasi, India
vipulg@cpan.org

Abstract

Hash Chains are used extensively in various cryptographic systems such as one-time passwords, server supported signatures, secure address resolution, certificate revocation and micropayments etc. However, currently they suffer from the limitation that they have a finite number of links which when exhausted requires the system to be re-initialized. In this paper, we present a new kind of hash chain which we call a Re-initializable Hash Chain (RHC). A RHC has the property that if its links are exhausted, it can be securely re-initialized in a non-repudiable manner to result in another RHC. This process can be continued indefinitely to give rise to an infinite length hash chain or more precisely an infinite number of finite length hash chains tied together. Finally we illustrate how a conventional hash chain (CHC) may be profitable replaced with RHC in cryptographic systems.

1 Introduction

The idea of one-way function chains or hash chains in short was first proposed by Lamport [1] to facilitate safeguarding one time password schemes (OTPs) from 'eavesdrop and replay' kinds of attack. Since then it has been employed in a wide range of applications. Hash chains have interesting public key cryptography like properties while employing nothing more than a fast one way hash function (OWHF).

Examples of system build upon the concept of hash chains include password based authentication [1, 3], certificate revocation [10, 37], secure address resolution [21], micropayments [19, 38, 39], online auctions [14], digital cash [40], secure server logs [22], efficient multicasting [11-13], server-supported signatures [35, 36], spam fighting protocols [41], one time signature schemes [42], sensor network security protocols[48-49] and securing routing information[43-47].

Most of these applications suffer from a common limitation, i.e. the hash chains used in these schemes have a finite length. To setup the system, the user chooses a constant say N to define the maximum number of operations to be allowed. Thereafter the system should be re-initialized. N cannot be made very large because of the increase in computational and storage requirements for the sender and also the verifier in many cases. Recently, a number of techniques for fast traversal [50-52] and efficient setup and verification [47, 48, 53] of hash chains were proposed. However, there is no technique, efficient or otherwise, for the re-initialization of hash chains. Addressing this is precisely the topic of this paper. We introduce the notion of 'tying' multiple finite length hash chains. This tying is achieved using one time signatures. This paper is also intended to serve as a comprehensive bibliography of the hash chain literature.

Rest of the paper is organized as follows- Section 2 quickly introduces the idea of hash chains. Section 3 discusses the related work. Section 4 provides a background on one time signatures. Section 5 introduces the proposed construction of RHCs. Section 6 gives examples about how the existing cryptographic systems employing hash chains may benefit from the concept of RHCs. Section 7 concludes the paper.

2 A Quick Introduction to Hash Chains

Hash chains are based a public function h that is easy to compute but computationally infeasible to invert, for suitable definitions of "easy" and "infeasible". Such functions are called one-way functions (OWF) and were first employed for use in login procedures by Needham [27]. If the output of a one-way function is of fixed length, it is called a one-way hash function (OWHF). More precisely, the definition of OWHF is given as [26]-

Definition A function h that maps bit strings, either of an arbitrary length or a predetermined length, to strings of a fixed length is a OWHF if it satisfies three additional properties-
→ Given x, it is easy to compute h(x)
→ Given h(x), it is hard to compute x
→ It is hard to find two values x and y such that h(x) = h(y), but $x \neq y$.

The idea of "hash chain" was first proposed by Lamport [1] in 1981 and suggested to be used for safeguarding against password eavesdropping. However being an elegant and versatile low-cost technique, the hash chain construction finds a lot of other applications.

A hash chain of length *N* is constructed by applying a one-way hash function *h*() recursively to an initial seed value s.

$$h^N(s) = h(h(h(...h(s)...))) \quad (N \text{ times})$$

The last element $h^N(s)$ also called the tip T of the hash chain resembles the public key in public key cryptography i.e., by knowing $h^N(s)$, $h^{N-1}(s)$ can not be generated by those who do not know the value *s*, however given $h^{N-1}(s)$, its correctness can be verified using $h^N(s)$. This property of hash chains has directly evolved from the property of one-way hash functions.

In most of the hash-chain applications, first $h^N(s)$ is securely distributed and then the elements of the hash chain are spent (or used) one by one by starting from $h^{N-1}(s)$ and continuing until the value of *s* is reached. At this point the hash chain is said to be exhausted and the whole process should be repeated again with a different s to reinitialize the systems. There are some other variants [47, 48] of this construction of hash chains. Our re-initialization technique applies to those constructions also.

3 Related Works

In an attempt to counter the 'limited-links' limitation of hash chains, Bicakci and Baykal [2] recently proposed the use of signature chains based on public key cryptography.

Construction: Let algorithm A be a public-key algorithm (e.g. RSA [20]) where d is the private key and e is the public key. Let s and c constitute a pair such that A(s, d) = c and A(c, e) = s
Note that $A^N(s, d)$ denotes that we apply the public key algorithm A recursively N times to the initial input (seed) s using the private key d.

As seen below, recursive applications results in an infinite length chain originated from the initial input s

S, A(s, d), A^2(s, d), . . . , A^{N-1}(s, d), A^N(s, d), . . .

However, in our view, using public key cryptography defeats the basic purpose of hash chains as it compromises on efficiency. Note that in most cases, hash chains are mainly used to remove/complement the usage of public key cryptography for gaining efficiency.

4 A Background on One Time Signatures

The Concept of One time signatures (OTS) has been known for over two decades. It was initially proposed by Lamport [17] and was the first digital signatures scheme ever designed. Interestingly, OTS schemes employ nothing more than OWHFs. The concept of OTS was subsequently enhanced by Merkle [28, 29], Winternitz [28] and Bicakci et al [33]. Bleichenbacher et al [30–32] formalized the concept of OTS using directed acyclic graphs (DAGs).

SIGNING A ONE BIT MESSAGE. The signer chooses as the secret key two values x1 and x2 (representing ‘0’ and ‘1’) and publishes their images under a one-way function y1 = h(x1) and y2 = h(x2) as the public key. These x’s and y’s are called the *secret key components* and the *public key components*, respectively. To sign a single bit message, reveal the pre-image corresponding to the actual ‘0’ or ‘1’ i.e. reveal x1 or x2 based upon whether the message to be signed is 0 or 1.

For signing longer messages, several instances of this basic scheme may be used. Thus we note that to sign an n bit message, 2n x’s and thus 2n y’s are required and the size of signatures generated is equal to n x’s i.e. n times the size of random number.

There are several improvements to this basic scheme. Merkle [28, 29] proposed an improvement which reduces the number of public as well as secret key components in the Lamport method by almost two-fold. Instead of generating two x’s and two y’s for each bit of the message, the signer generates only one x and one y for each bit of the message to be signed. When one of the bits in the message to be signed is a ‘1’, the signer releases the corresponding value of x; but when the bit to be signed is a ‘0’, the signer releases nothing. Because this allows the receiver to pretend that he did not receive some of the x’s, and therefore to pretend that some of the ‘1’ bits in the signed message were ‘0’, the signer must also sign count of the ‘0’ bits in the message. Now, when the receiver pretends that a ‘1’ bit was actually a ‘0’ bit, he must also increase the value of the count field, which can’t be done. Because the count field has only $\log_2(n)$ bits in it, the number of public and secret key components is decreased by almost a factor of two i.e. from 2n to $n + \log_2(n)$ (or to $n + [\log_2(n)] + 1$ if n is not a power of 2). This also results in the decrease of signature size by almost a factor of two.

As an example, if we wished to sign the 8-bit message ‘0100 1110’ we would first count the number of ‘0’ bits (there are 4) and then append a 3-bit count field (with the value 4) to the original 8-bit message producing the 11-bit message ‘0100 1110 100’which we would sign by releasing x[2], x[5], x[6], x[7] and x[9]. The receiver cannot pretend that he did not receive x[2], because the resulting erroneous message ‘0000 1110 100’ would have 5 ‘0’s in it, not 4.
Similarly, pretending he did not receive x[9] would produce the erroneous message ‘0100 1110 000’ in which the count field indicates that there should be no ‘0’s at all. There is no combination

of x's that the receiver could pretend not to have received that would let him concoct a legitimate message.

Winternitz [28] proposed an improvement which reduces the signature size by several folds at the expense of increased computational effort. In Winternitz's method, the OWF is applied to two secret key components iteratively for a fixed number of times, resulting in a two-component public key.

Even though OTS are significantly efficient and faster than public key digital signatures, their use as full fledged signatures schemes is limited due to their inherent one time ness. However we note that for our purpose, they can prove to be a good tool because a hash chain only needs to be re-initialized once (which would subsequently result in another re-initializable hash chain and so on). OTS have also found some other good uses like online/offline signatures [24] where signing a message has two phases. The first and the more expensive one is performed offline before the message to be signed is even know. This phase is signing the public key components of a one time signature scheme instance using conventional signature schemes such as RSA. The second and much faster phase is signing the actual message using OTS.

Before going further, we introduce some basic notations used in this paper-

$l \rightarrow$ length of the output of OWHF employed e.g. 128 bit for MD5
$m \rightarrow$ number of public/secret key components used in the OTS scheme. Equal to $l + \log_2(l)$ for Merkle's construction
$P_U \rightarrow$ one time public key of user U. Equal to the collection (or Concatenation) of m public key components.
$S_U \rightarrow$ one time secret key of user U. Equal to the collection (or Concatenation) of m secret key components.

5 The Proposed Construction

The construction of a Re-initializable Hash Chain (RHC) is similar to that of a Conventional Hash Chain (CHC) except for the tip T. Recall that the tip T of a hash chain is computed in the last.

The user U first chooses a seed s either randomly or pseudo randomly and computes a CHC of length N-1

$$h^{N-1}(s) = h(h(h(...h(s)...))) \quad (N-1 \text{ times})$$

Now, U generates an instance of the OTS scheme by generating S_U and P_U and securely stores S_U (Note that this requirement may be relaxed if S_U is generated pseudo randomly)

Finally, the tip T of the hash chain is computed as follows

$$T = h(h^{N-1}(s), h(P_U))$$

Now, T is the tip of a RHC of length N. T is made public/distributed to the appropriate parties depending upon the application in which it is used.

For using the first link of the constructed RHC, U sends $h^{N-1}(s)$ as well as $h(P_U)$ which the verifier(s) can verify using the tip T. Alternatively, $h(P_U)$ can be distributed in the beginning along with the tip of the chain. In that case, U needs to only send $h^{N-1}(s)$ as the first link of the

chain in the usual manner. Rest of the links in the chain are computed as for CHCs i.e. the i^{th} link is computed as $h^{N-i}(s)$. U can keep spending the links till i reaches 0, i.e. $h^0(s)$ or s is spent. At this point, the RHC is exhausted and a new RHC should be computed and tied to the existing RHC. For this purpose, we use the previously generated instance of the OTS scheme.

U choose a new seed s' and generates another instance of the OTS scheme by generating S_U' and P_U'. The tip T' of the new RHC is computed in the similar manner as before-

$$T' = h(h^{N-1}(s'), h(P_U'))$$

U now sends, in addition to T', the public key instance P_U and the appropriate secret key components from S_U required to sign T' to the verifier. The verifier checks the sent public key P_U using $h(P_U)$ embedded in the previous RHC, checks the signature on T' using the revealed components of S_U and accepts the new RHC tip T'. At this point, the user U is ready to spend the links of the new RHC with tip T'. This process can continue indefinitely in the similar way to result in an infinite number of finite length hash chains tied together. Note that non-repudiability is maintained throughout.

Improving the Efficiency of the Basic Scheme

The efficiency of the described construction can be improved if the to be revealed components of the secret key are also 'spent' in the same manner as the normal links of the RHC.

Consider that in the OTS of the tip T' of the new RHC, there are n bits whose value is 1. Then using the improved scheme, a RHC having N links may be reusable (N+n+1) times. In the average case, assuming that half of the m (= $l + \log_2 l$) bits in the OTS would be 1, a RHC of length N may be used (N + m/2 + 1) times. This can be done as follows-

a) The N links of the RHC are spent in the usual manner.
b) After N spendings, U gives out the public key P_U as one more link which can be verified using $h(P_U)$.
c) After this, U computes T' and supplies T' to the verifier which stores it without verification. Now U gives out the secret key components required to sign T's one by one as links. Thus considering the average case, m/2 more spendings can be done.

After all the required secret key components are revealed, the verifier would have authenticated T's and thus the user U is now ready to spend RHC with tip T'. Again note that non-repudiability is maintained throughout.

6 Example Schemes Replacing CHCs with RHCs

Now we move on to examples schemes demonstrating the profitable replacement of CHCs with RHCs.

6.1 One Time Passwords (OTPs)

Introduced in 1981 by Lamport [1] and subsequently implemented, improved [3, 25] and standardized in several rfcs [6-8], OTPs are considered as a popular choice for password based authentication. OTPs are more commonly known as Lamport hashes or S/KEY™ One Time Password System. Currently, they suffer from the limitation that a user may only authenticate a finite number of times say N before the system should be re-initialized. The re-initialization can either be done manually or automatically. Automatic re-initializations suffer from straightforward

active attacks in which an adversary may take complete control of the user account. Further, the value of N should also be kept low since the computation burden per authentication on the client increases linearly with N. The system can be substantially improved by employing our concept of RHCs. This is despite the fact that in this environment, the user and the generator of hash chain is a human remembering only a password and no secret key components. Relaxing the client side storage requirement is accomplished by generating the secret key components pseudo randomly using the password as the seed to the pseudorandom function. The system is worth giving a special treatment. We extend the idea of OTPs using RHCs in a forthcoming work. An interesting point in the paper is the construction of a somewhat more efficient method of re-initializing hash chains without offering non-repudiability and not using OTS. This construction is usable in OTPs because contrary to most other schemes employing hash chains, OTPs do not require non-repudiation.

6.2 Micropayment Schemes

To support micropayments, exceptional efficiency is required; otherwise the cost of the mechanism will exceed the value of the payments. This directly implies that the number of public key operations should be minimized. As a consequence, micropayment schemes using hash chains were developed [19]. Very informally, the basic idea is that the user (or the customer), for repeated payments of small and fixed amount to the merchant, generates a hash chain and digitally signs its tip along with other information such as the merchant identifier and the value of each payment. The customer passes the signed message to the merchant. Now, the customer can pay the fixed amount to the merchant anytime by releasing a link of the hash chain. Multiples of this amount may also be paid by releasing multiple links.

A direct advantage to the merchant is that now she can aggregate many payments from a single customer and get the payments credited to her account in a single transaction with the bank (recall that the banks usually charge a transaction fee from the merchant for every transaction). This is done by sending the signed message received from the customer containing the tip $h^N(s)$ of the hash chain along with the value of the last link $h^{N-i}(s)$ collected from the customer to the bank. Thus the bank transfers the amount i*c to the merchant where c is the value of each payment as specified in the signature. Further, the system dramatically reduces the required computation for all the 3 parties involved in the system. This also makes it feasible for the customer to access the service of merchant on low power mobile devices where public key cryptography and hence conventional payments are not possible. For more details about such micropayment systems, an interested reader is referred to payword [19].

A problem however with the system is that the customer should again sign a new message containing the tip of a new hash chain once all the links in the hash chain are exhausted. This can be overcome by using RHCs where the customer signs the tip of a RHC instead of a CHC in the beginning. Thus the customer can now continue payments indefinitely without the need of generating a new digital signature after some time. This results in better payment aggregation for the merchant as well as the feasibility of continual mobile device usage for the customer. Note that the system is exceptionally efficient especially when the improved version of RHCs is used.

6.3 Server Supported Signatures

Server supported signature schemes allow a user using a constrained mobile device to digitally sign a message by employing a semi trusted server called a virtual server (VS). An example is the SAS protocol [35, 36]. First we provide a brief summary of the SAS protocol, for details, the reader is referred to [35].

There is an initialization phase in SAS where each user (originator) gets a certificate from an offline certification authority specifying $h^N(s)$, the tip of the hash chain, where s is kept secret by the originator O. In addition, O should register to a VS (which has the traditional public-key based signing capability) before operation. Then the SAS protocol works in three rounds-

1. The originator (O) sends m and $h^{N-i}(s)$ to VS where
– m is the message
– $h^{N-i}(s)$ is the i^{th} element of the hash chain. The counter i is initially set to 1 and incremented after each run.

2. Having received O's request, VS checks the followings:
– Whether O's certificate is revoked or not.
– Whether h^i(supplied i^{th} link) = $h^N(s)$ or in a more efficient way h(supplied i^{th} link) = $h^{N-i+1}(s)$ since $h^{N-i+1}(s)$ has already been received as the $(i-1)^{th}$ link
If these checks are OK, VS signs m concatenated with $h^{N-i}(s)$ and sends it back to *O*.

3. After receiving the signed message from VS, O verifies the VS's signature, attaches $h^{N-i-1}(s)$ to this message and sends it to the receiver R.
Upon receipt of the signed message, the receiver verifies VS's signature and checks whether $h(h^{N-i-1}(s)) = h^{N-i}(s)$

Note that VS may try to sign m' instead of m once O releases the i^{th} link. But then, since O verifies the signature, it would not release the $(i-1)^{th}$ link thus rendering the signature on m' incomplete. The best VS can do is to sign two messages m and m' with the same link embedded in both signatures. However, this leaves O with a cryptographic proof of server fraud if she gets hold of both signatures. [35] discusses efficient techniques to safeguard O against such VS frauds.

It is clear that once all the links of the hash chain are exhausted, the originator O should generate another hash chain and get a new certificate from the CA specifying its tip. Thus, the system can be significantly improved by employing RHCs instead of the CHCs. The CA would certify the tip of a RHC generated by the originator O. O spends the links in the RHC. Once all the links are exhausted, a new RHC may be computed and tied to the existing one. Thus the system can continue without the need to get new certificate from the CA. This may also be a cost saving for O since presumably the CA would charge an amount to issue a new certificate for O.

Thus we see from the above illustrative examples that CHCs can be quite easily replaced by RHCs in various cryptographic systems. As the examples demonstrate, the details of modifying a system to use RHCs instead of CHCs are quite straight forward. Thus, we leave it as future work to apply the idea of RHCs in various other systems employing hash chains.

7 Conclusions

The need for an elegant method for the secure re-initialization of hash chains was clear due to the 'limited-link' limitations most systems employing hash chains suffer from. In response, we have presented a new kind of hash chains which we call RHCs. A RHC has the property that if its links are exhausted, it can be securely re-initialized in a non-repudiable manner to result in another RHC. This process can be continued indefinitely to give rise to an infinite number of finite length hash chains tied together.

We replaced CHCs with RHCs in OTPs, micropayment systems and server supported signature schemes and demonstrate that the modified systems overcomes the 'limited-link' limitation of the original systems. We believe that RHCs may be profitably employed in a similar manner in other systems as well.

References

[1] L. Lamport, "Password Authentication with Insecure Communication", Communications of the ACM 24.11 (November 1981), pp 770-772.

[2] K. Bicakci and N. Baykal, Infinite Length Hash Chains and Their Applications, Proceedings of IEEE 11th International Workshops on Enabling Technologies (WETICE 2002), June 10-12, 2002, Pittsburgh, USA.

[3] N Haller, "The S/KEY One-Time Password System", Proceedings of the ISOC Symposium on Network and Distributed System Security, pp 151-157, February 1994.

[4] G Tsudik, "Message authentication with one-way hash functions", ACM Computer Communications Review, 22(5), 1992, pp 29-38.

[5] A. D. Rubin, "Independent One-Time Passwords", USENIX Journal of Computer Systems, February 1996.

[6] N Haller, "The S/KEY One-Time Password System", RFC 1760, February 1995. Available from http://www.ietf.org.

[7] N Haller, "A One-Time Password System", RFC 1938, May 1996. Available from http://www.ietf.org.

[8] N Haller, C. Metz, P. Nesser and M. Straw, "A One-Time Password System", RFC 2289, Feb 1998. Available from http://www.ietf.org.

[9] National Institute of Standards and Technology (NIST), "Announcing the Secure Hash Standard", FIPS 180-1, U.S. Department of Commerce, April 1995.

[10] S. Micali, "Eficient Certificate Revocation," Proceedings of RSA '97, and U.S. Patent No. 5,666,416.

[11] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Eficient and Secure Source Authentication for Multicast," Proceedings of Network and Distributed System Security Symposium NDSS 2001, February 2001.

[12] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Eficient Authentication and Signing of Multicast Streams over Lossy Channels," Proc. of IEEE Security and Privacy Symposium S & P 2000, May 2000.

[13] A. Perrig, R. Canetti, D. Song, and D. Tygar, "TESLA: Multicast Source Authentication Transform", Proposed IRTF draft, http://paris.cs.berkeley.edu/ ~perrig/

[14] S. Stubblebine and P. Syverson, "Fair On-line Auctions Without Special Trusted Parties," Financial Cryptography '01.

[15] R Impagliazzo and S Rudich, "Limites on the Provable Consequances of one way permutations," Proceedings of the 21st Annual Symposium on Theory of Computing, ACM, 1989.

[16] S. Halevi and H. Krawczyk, "Public-Key Cryptography and Password Protocols," ACM Transactions on Information and System Security, Vol. 2, No. 3, August 1999, Pages 230–268.

[17] W. Diffie, and M. E. Hellman, "New directions in cryptography". IEEE Transactions on Information Theory IT-11 (November 1976), 644–654.

[18] R. H. Morris, and K. Thompson, "Unix password security", Communications of the ACM 22, 11 (November 1979), 594.

[19] R. L. Rivest and A. Shamir. PayWord and MicroMint-two simple micropayment schemes. In Mark Lomas, editor, Proceedings of 1996 International Workshop on

Security Protocols, volume 1189, Lecture Notes in Computer Science, pages 69-87. Springer, 1997.
[20] R. L. Rivest, A. Shamir, and L. M. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, 21(2):120–126, 1978.
[21] V. Goyal, “A Solution to the ARP Cache Poisoning Problem”, manuscript, April 2004.
[22] B. Schneier and J. Kelsey, “Cryptographic support for secure logs on untrusted machines”, In Proceedings 7th USENIX Security Symposium (San Antonio, Texas), Jan 1998.
[23] P. T. Devanbu and S. Stubblebine. Stack and queue integrity on hostile platforms. In Proceedings 1998 IEEE Symposium on Research in Security and Privacy, May 1998. (Oakland).
[24] S. Even, O. Goldreich, and S. Micali, “On-line/off-line digital signatures”, Crypto ‘89, LNCS Vol No. 435, pp 263–277, Springer-Verlag, 1990.
[25] D.L. McDonald, R.J. Atkinson, C. Metz "One-Time Passwords in Everything (OPIE): Experiences with Building and Using Strong Authentication," In Proc. of the 5th USENIX UNIX Security Symposium, June 1995.
[26] T.A. Berson, L. Gong and T.M.A. Lomas, Secure, “Keyed, and Collisionful Hash Functions, Technical Report” SRI-CSL-94-08, SRI International, May 1994.
[27] M. V. Wilkes, “Time-Sharing Computer Systems”, New York: Elsevier, 1972.
[28] R.C. Merkle, A Digital Signature Based on a Conventional Encryption Function, Proc. CRYPTO’87, LNCS 293, Springer Verlag, 1987, pp 369-378.
[29] R.C. Merkle, A Certified Digital Signature, Proc. CRYPTO’89, LNCS 435, Springer Verlag, 1990, pp 218-238.
[30] D. Bleichenbacher and U.M. Maurer, Directed Acyclic Graphs, One-way Functions and Digital Signatures, Proc. CRYPTO’94, LNCS 839, Springer Verlag, 1994, pp 75-82.
[31] D. Bleichenbacher, U.M. Maurer, Optimal Tree-Based One-time Digital Signature Schemes, Proc. STACS’96, LNCS 1046, Springer-Verlag, pages: 363-374, 1996.
[32] D. Bleichenbacher, U.M. Maurer, On the efficiency of one-time digital signatures, Proc. ASIACRYPT’96, LNCS 1163. Springer-Verlag, pages: 145-158, 1996.
[33] K. Bicakci, G. Tsudik, B. Tung, How to construct optimal one-time signatures, Computer Networks (Elsevier), Vol.43(3), pp. 339-349, October 2003.
[34] M.Jakobsson and S.Wetzel, Secure Server-Aided Signature Generation. In Proc.of the Workshop on Practice and Theory in Public Key Cryptography (PKC 2001), LNCS No.1992, Springer, 2001.
[35] N.Asokan, G.Tsudik and M.Waidners, “Server-supported signatures”, Journal of Computer Security, November 1997.
[36] X.Ding, D.Mazzocchi and G.Tsudik. Experimenting with Server-Aided Signatures, Network and Distributed Systems Security Symposium (NDSS ’02), February 2002.
[37] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation. In Hugo Krawczyk, editor, Advances in Cryptology – Crypto ’98, volume 1462 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 1998.
[38] Ralf Hauser, Michael Steiner, and Michael Waidner. Micro-payments based on iKP. In 14th Worldwide Congress on Computer and Communications Security Protection, pages 67–82, C.N.I.T Paris-La Defense, France, June 1996.
[39] Torben Pryds Pedersen. Electronic payments of small amounts. Mark Lomas, editor. Security Protocols—International Workshop, volume 1189 of Lecture Notes in Computer Science, Cambridge, UK, April 1997. Springer-Verlag, Berlin Germany.
[40] Khanh Quoc Nguyen, Yi Mu, Vijay Varadharajan, “Digital Coins based on Hash Chain”.

[41] C. Dwork, A. Goldberg, and M. Naor. On Memory-Bound Funtions for Fighting Spam. Advances in Cryptology - Crypto 2003, Volume 2729 of Lecture Notes in Computer Science. Springer-Verlag, 2003.
[42] A. Perrig. The BiBa One-Time Signature and Broadcast Authentication Protocol. Proceedings of the Annual Conference on Computer and Communications Security (CCS), pages 28-37. ACM Press, 2001.
[43] Steven Cheung. An efficient message authentication scheme for link state routing. In 13th Annual Computer Security Applications Conference, pages 90–98, 1997.
[44] Ralf Hauser, Antoni Przygienda, and Gene Tsudik. Reducing the cost of security in link state routing. In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '97), pages 93–99, San Diego, California, February 1997. Internet Society.
[45] Kan Zhang. Efficient protocols for signing routing messages. In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '98), San Diego, California, March 1998. Internet Society.
[46] Y.-C. Hu, D. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. Workshop on Mobile Computing Systems and Applications (WMCSA) 2002. IEEE Computer Society Press, 2002.
[47] Y.-C. Hu, A. Perrig, and D. Johnson. Efficient Security Mechanisms for Routing Protocols. Annual Symposium on Network and Distributed System Security (NDSS) 2003. Internet Society, 2003.
[48] Donggang Liu and Peng Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In Network and Distributed System Security Symposium, NDSS '03, pages 263– 276, February 2003.
[49] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. Wireless Networks, 8(5):521–534, September 2002.
[50] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In Proceedings of the Fourth Conference on Financial Cryptography (FC '02), Lecture Notes in Computer Science, 2002.
[51] M. Jakobsson. Fractal hash sequence representation and traversal. In Proceedings of the 2002 IEEE International Symposium on Information Theory (ISIT '02), pages 437–444, July 2002.
[52] Yaron Sella. On the computation-storage trade-offs of hash chain traversal. In Proceedings of Financial Cryptography 2003 (FC 2003), 2003.
[53] Marc Fischlin. Fast Verification of Hash Chains. In Proceedings of RSA Security 2004 - Cryptographer's Track, Lecture Notes in Computer Science, Springer-Verlag, 2004.